Universitat de Lleida

Escola Politècnica Superior

Enginyeria Tècnica en Informàtica de Sistemes/Gestió

Treball de final de carrera

Aplicació web per a la Gestió de Llicències de la Federació Catalana de Piragüisme

Autors: Víctor Mateu Messeguer i M.Àngels Cerveró Abelló

Director: Josep Maria Ribó Balust

Setembre del 2006

Aplicació web per a la Gestió de Llicències de la Federació Catalana de Piragüisme

Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Gestió

Autors: Víctor Mateu Messeguer i M.Àngels Cerveró Abelló

Director: Dr. Josep Maria Ribó Balust

Agraïments

En primer lloc, agrair l'esforç, l'ajut, el seguiment, els consells i l'encaminament realitzats pel director del projecte, Dr. Josep Maria Ribó.

Fer especial menció als treballadors de la Federació Catalana de Piragüisme, senyores i senyors:

.- Pepita Fileia Herta
.- Carme Adell Argilés
.- Dr. Joan Ignasi Rosell Urrutia
.- German Rei Romero

A tots ells dedicar-los un afectuós agraïment i reconeixement per tota l'ajuda donada, per les hores que ens han tingut al seu lloc de treball explicant-nos com funcionava tot el sistema de gestió de llicències, per la seva amabilitat i per la paciència que han tingut.

No volem deixar-nos als nostres companys més propers amb els que hem patit conjuntament les tasques de realitzar un projecte final de carrera, ells són en Jordi Monné, en Jesús Ojeda i la Míriam Clavé. Aquest projecte no seria el mateix si no fora per la seva col·laboració. I ja per finalitzar un agraïment general a tota la gent, amics, en especial a la Natàlia i al Valentín, i familiars que ens han aguantat en els nostres moments d'estrès generalitzat. A tots ells, gràcies per la seva col·laboració.

Índex

Conté CD a la contraportada amb l'aplicació i l'API realitzats

Capítol 1

Introducció

La Federació Catalana de Piragüisme, constituïda com a tal l'any 1954 i amb seu a Lleida, té com a principal objectiu promoure la pràctica del piragüisme amb tot el que això comporta:

- Organització de competicions.
- Preparació de tècnics especialitzats en aquest esport i en les seves diferents manifestacions.
- Gestionar les llicències que donen el privilegi de participar en les competicions i proporcionen una assegurança als federats.

Justament en aquest últim punt, en la gestió de llicències, és on pren protagonisme el projecte de final de carrera que es presenta en aquesta documentació.

Aquesta gestió de llicències, no únicament significa la donada d'alta d'un federat, sinó també el posterior seguiment d'aquest, entenen com a seguiment, la possible renovació un cop la llicència ha caducat, la modificació i consulta de dades i la realització de cerques amb finalitats estadístiques i/o informatives per als membres directius i col·laboradors de l'entitat catalana.

No obstant, actualment la Federació Catalana de Piragüisme ja disposa d'una aplicació d'escriptori amb la funcionalitat desitjada. Això implica que el control sobre les llicències i tota la feina que això comporta, recaigui absolutament sobre els treballadors de la FCP. Per tal de permetre una alliberació de feina i una major interacció entre la FCP i els clubs que la componen, s'ha realitzat el treball de final de carrera "Aplicació Web per a la Gestió de les Llicències de la FCP".

1.1 Objectius

- Disseny i implementació d'una aplicació web, sota una arquitectura "Model-Vista-Controlador", que permeti la gestió de llicències de la FCP de tal manera que se n'obtinguin dues conseqüències:
    a. Substitució de l'actual aplicació d'escriptori amb la qual es realitza el control de les llicències.
    b. Permetre que els clubs tinguin participació activa sobre el control de les llicències de federat dels seus socis agilitant, d'aquesta manera, les tasques que fins ara realitzava íntegrament la FCP.
- En cap moment es pretén que l'aplicació creada prengui el protagonisme a la FCP. Això significa que, tot i que els clubs tinguin la capacitat de gestionar als seus socis com a federats, la última paraula sobre la validació de llicències i de dades personals de tota persona o entitat vinculada a la federació la tindrà la pròpia federació.

1.2 Contingut de la memòria

Aquesta memòria està dividida en 6 capítols, els continguts resumits dels quals s'expliquen a continuació:

Capítol 1: Introducció
Capítol actual. Pretén fer una breu presentació de la memòria i dels objectius del treball final de carrera.

Capítol 2: Anàlisi de Requeriments
En aquest capítol s'explica com s'ha treballat amb els usuaris per tal d'obtenir tots els requeriments que l'aplicació web havia de complir.
Tots aquest requeriments es troben completament descrits ens els respectius casos d'ús.

Capítol 3: Model de Domini
En aquest capítol s'explica el model de domini sorgit a partir dels requeriments demanats i pactats amb els usuaris.

Capítol 4: Disseny
En aquest capítol s'ha intentat explicar el disseny seguit per tal de realitzar la implementació de l'aplicació. S'han comentat les característiques del projecte a nivell d'arquitectura, dades, patrons i tecnologies utilitzades.

Capítol 5: Pressupost
Al tractar-se d'un treball que es podria qualificar de real, ja que l'aplicació ha estat realitzada per uns usuaris concrets i l'objectiu final és que es pugui utilitzar en l'àmbit de la federació, s'ha realitzat un petit pressupost les característiques del qual queden exposades en aquest capítol.

Capítol 6: Conclusions i Treball Futur
L'últim capítol de la memòria està dedicat a realitzar les conclusions del tot el treball fet i a exposar les tasques i ampliacions futures que es poden realitzar sobre el projecte actual.

Annex A: Ajax i javascript
En aquest annex es pretén realitzar una breu introducció a la tecnologia Ajax que s'ha utilitzat per la implementació del projecte passant, també, per una explicació sobre javascript.

Annex B: Framework Struts
En aquest capítol s'ha realitzat una explicació, amb intents pedagògics, sobre el framework utilitzat en la implementació del codi de l'aplicació, el qual atorga al projecte una arquitectura "Model – Vista – Controlador".

Annex C: Llicència
En aquest annex es presenta la llicència CC-GPL sota la qual es vol presentar el treball realitzat.

Annex D: Mostra de l'API
En aquest annex es mostren uns petits exemples de l'API de l'aplicació realitzat.

Capítol 2

Anàlisi dels requeriments

2.1 Introducció

Tot projecte es caracteritza pel fet que ha de complir uns objectius, és a dir, ha de permetre a un grup d'usuaris realitzar un conjunt de tasques. Aquests objectius o tasques constitueixen el que s'anomena "requeriments del sistema".
La realització d'un bon anàlisi dels requeriments permetrà als desenvolupadors conèixer les necessitats dels futur usuaris del sistema i aprendre el funcionament del medi en el qual haurà de treballar aquest[1]. Aquests coneixements permetran adaptar l'aplicació als desitjos dels usuaris i minimitzar-ne els possibles errors que puguin aparèixer.

En el cas del treball que s'està documentant en aquestes pàgines, es tracta d'un projecte real, és a dir, s'ha treballat amb usuaris externs a la universitat. Concretant una mica més, els usuaris que ens han ajudat a establir els requeriments del sistema han estat els treballadors de la Federació Catalana de Piragüisme (FCP). Això ha provocat el fet que es realitzessin diferents reunions entre els integrants del grup de treball i el personal de la FCP, en les quals s'han anat concretant i definint les funcionalitats que hauria de tenir el sistema i la manera de treballar i funcionar que els usuaris tenen actualment en el seu medi laboral.

En les pròximes pàgines es pretén reflectir el contingut i les conclusions a les quals es va arribar a cadascuna de les trobades realitzades. Finalment, s'indicaran quins seran els usuaris finals de l'aplicació i s'exposaran els casos d'ús de totes les funcionalitats del projecte realitzat.

[1] Per exemple, aprendre la gestió i distribució dels horaris d'una empresa per la qual s'ha d'elaborar un projecte que permeti flexibilitzar les hores de treball als empleats.

2.2 Reunions amb els usuaris

Al llarg de tot el projecte s'han realitzar quatre reunions amb els usuaris tal per establir els requeriments. Els continguts i les conclusions a les quals es va arribar en cadascuna d'elles es presentaran a continuació.

2.2.1 Reunió Inicial

Els participants en aquesta reunió van ser els següents:

1. Director del Projecte: Dr. Josep Maria Ribó
2. Alumnes que realitzen el Projecte: Víctor Mateu i M.Àngels Cerveró
3. Usuaris de la FCP, senyors: Joan Ignasi Rosell i German Rei

Primeres idees que els usuaris van donar sobre l'aplicació a realitzar:

El projecte pel qual estan interessats els usuaris té dues parts ben diferenciades:

a. Gestió de les llicències dels federats
b. Gestió de les competicions

La primera part es basa en el desig de realitzar les altes, validacions i renovacions de federats i modificacions i consultes de les dades dels mateixos via web per tal que tant els treballadors de la FCP, com els dels clubs i els mateixos federats hi puguin accedir amb facilitat.

En un primer moment, ens indiquen que de federats en pot haver de 4 tipus:

a. Esportistes
b. Àrbitres
c. Tècnics
d. Directius

Tots ells pertanyents a un club determinat que ha de tenir llicència[2] de la federació, és a dir, que ha hagut de pagar per tal de federar-se ell mateix i poder federar als seus socis.

A part, també poden tenir llicència de la FCP les empreses o negocis que ho demanin i paguin.

Pel que fa a la gestió de les competicions, els usuaris pretenen controlar informàticament quins federats poden participar (ja sigui arbitrant, impartint conceptes o practicant el piragüisme) a quines competicions segons les seves característiques com a àrbitres, tècnics o esportistes.

Conclusions a les quals es va arribar:

Tenint una petita idea del que cal fer, s'acota el domini del projecte a la gestió de les llicències dels federats. Es decideix, per tant, que la gestió de les competicions serà objecte d'un treball de final de carrera futur.

[2] Tenir llicència de la FCP o estar federat significa que la federació demana una assegurança per cobrir els possibles accidents que es puguin tenir practicant piragüisme i, a més a més, dóna dret a participar a les competicions que es convoquin durant el període en que la llicència és vigent.

2.2.2 Reunions amb els usuaris dels dies 20-03-2006 i 03-04-2006

Els participants en aquestes reunions van ser els següents:

1. Alumnes que realitzen el Projecte: Víctor Mateu i M.Àngels Cerveró
2. Usuaris de la FCP, senyora i senyor: Pepita Fileia i German Rei

Primeres idees que els usuaris van donar sobre la gestió de les llicències:

Es pretén dur a terme l'alta dels federats, els quals es poden dividir en:

a. Entitats: clubs
b. Federats: tècnics, àrbitres i esportistes (tots han de ser socis d'un club).

L'alta significa la introducció de les dades personals i de llicència del federat en una base de dades[3].

D'altra banda, també es desitja poder modificar aquestes dades i consultar-les.

Pel que fa a permetre l'accés a l'aplicació tant a Clubs com a Federats, els treballadors prefereixen que només puguin ser els clubs els que puguin tenir control sobre les dades dels seus socis, de tal manera, que els federats no hi podran accedir ni tenir cap tipus de privilegi dins l'aplicació a crear.

A més a més, qualsevol canvi o alta que realitzi un club s'haurà de notificar a la FCP.

Una funcionalitat addicional serà la creació de fitxers xls:

a. Amb les dades de tots els federats que s'han donat d'alta en el dia actual per tal de passar-ho a l'asseguradora.
b. Amb les dades de les consultes realitzades pels treballadors de la federació.

Conclusions a les quals es va arribar:

Les funcionalitats a realitzar són les següents:

a. Alta i renovació, des de la FCP, de clubs, esportistes, àrbitres i tècnics
b. Modificació i consulta, des de la FCP, de dades de clubs, esportistes, àrbitres i tècnics.
c. Alta, des del Club, d'esportistes, àrbitres i tècnics
d. Renovació, des del club, del propi club, esportistes, àrbitres i tècnics.
e. Validació de les dades, des de la FCP, que han estat inserides o modificades per un determinat club.
f. Creació del fitxer de la mútua asseguradora
g. Creació del fitxer que reflecteixi les dades obtingudes a partir d'una consulta.

[3] Es podrà veure quines són aquestes dades ens els posteriors casos d'ús i en el model de domini.

2.2.3 Reunió amb els usuaris del dia 09-08-2006

Els participants en aquesta reunions van ser els següents:

1. Alumnes que realitzen el Projecte: Víctor Mateu i M.Àngels Cerveró
2. Usuaris de la FCP, senyores: Pepita Fileia i Carme Adell

Desenvolupament de la reunió:

Es va presentar una primera versió funcional del projecte amb gran part de les funcionalitats pactades realitzades.

Els usuaris van explicar que actualment la federació està patint canvis en la seva administració i gestió, la qual cosa implica un canvi en els seus estatuts i manera de treballar.

A més a més, també ens van demanar l'addició de noves funcionalitats, les quals s'expliquen a continuació:

a. Alta, consulta, modificació i renovació de Directius i Empreses
b. Realització d'un fitxer xls amb les dades que es presenten en el carnet de federat per tal de poder-lo imprimir i crear amb més facilitat el document de la llicència.
c. Realització de la memòria de l'any passat, és a dir, creació d'un fitxer xls amb les dades de totes les entitats i persones que van estar federades l'any anterior.

Conclusions a les quals es va arribar:

El treball quedarà acotat a la realització de les funcionalitats ja pactades en reunions anteriors i, a més a més s'hi afegiran les següents capacitats:

a. Alta i renovació, des de la FCP, de directius.
b. Modificació i consulta, des de la FCP, de directius
c. Alta i renovació, des del Club, de directius
d. Validació de les dades dels directius, des de la FCP, que han estat inserides o modificades per un determinat club.
e. Creació d'un fitxer xls amb la memòria de l'any anterior.

Es deixa com ampliació futura la incorporació d'empreses com a entitats federades i la realització del fitxer que permeti crear els carnets de federat.

2.3 Usuaris de l'aplicació

L'aplicació per a la Gestió de les Llicències de la FCP té dos tipus d'usuaris:

a. Treballadors de la Federació
b. Treballadors dels Clubs

A continuació s'explicaran les principals característiques de cadascun d'ells i les tasques que poden realitzar.

2.3.1 Treballadors de la Federació

Els treballadors de la Federació tenen control complet sobre l'aplicació i les tasques que poden realitzar són les següents:

a. Alta de federats:
   - Clubs
   - Esportistes
   - Àrbitres
   - Tècnics
   - Directius
b. Modificació i consulta de les dades privades i personals dels federats
   - Clubs
   - Esportistes
   - Àrbitres
   - Tècnics
   - Directius
c. Validació de la llicència
   - Clubs
   - Esportistes
   - Àrbitres
   - Tècnics
   - Directius
d. Validació de les dades
   - Clubs
   - Esportistes
   - Àrbitres
   - Tècnics
   - Directius
e. Renovació de la llicència
   - Clubs
   - Esportistes
   - Àrbitres
   - Tècnics
   - Directius
f. Obtenció de les dades per l'asseguradora
   - Esportistes
   - Àrbitres

    - Tècnics
    - Directius
g. Realització de cerques generals de federats
h. Consulta de les categories d'esportistes existents

Finalment, dir que els treballadors de la FCP accediran a l'aplicació a gràcies a un nom d'usuari i una contrasenya

2.3.2 Treballadors dels clubs

Els treballadors dels Clubs tenen menys privilegis que els treballadors de la federació i les seves tasques es limiten a les següents:

a. Alta de federats que siguin socis del club:
    - Esportistes
    - Àrbitres
    - Tècnics
    - Directius
b. Modificació i consulta de les dades privades i personals dels federats socis del club
    - El propi club
    - Esportistes
    - Àrbitres
    - Tècnics
    - Directius
c. Renovació de la llicència dels federats socis del club
    - El propi club
    - Esportistes
    - Àrbitres
    - Tècnics
    - Directius
d. Realització de cerques generals de federats socis del club.

Finalment, dir que els treballadors dels Clubs accediran a l'aplicació a gràcies a un nom d'usuari i una contrasenya que els hauran estat donats en el moment de donar-se d'alta com a federats.

2.4 Casos d'Ús

En les pròximes pàgines es presentaran els casos d'ús de les funcionalitats del sistema. S'intentarà unificar les funcionalitats per a tots els federats en conjunt per tal que aquest apartat no es faci pesat i repetitiu. No obstant, els casos d'ús de "Donar d'alta" es faran per cada tipus de federat (club, esportista, àrbitre, tècnic i directiu), ja que d'aquesta manera s'aprofitarà per indicar quines dades caracteritzaran cadascun d'ells.

2.4.1 Autenticació:

- **Cas d'ús**: Autenticació d'un usuari
- **Actors**
    1. Usuari
- **Propòsit**
  Autenticar l'usuari per tal que pugui accedir a l'aplicació per a la gestió de llicències
- **Resum**
  Un usuari interessat en accedir a l'aplicació per a la gestió de llicències s'autenticarà i identificarà.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un usuari interessat en accedir a l'aplicació per a la gestió de llicències introdueix el seu nom d'usuari i contrasenya per tal d'autenticar-se i identificar-se	2.- El sistema comprovarà que l'usuari identificat pel nom d'usuari pertany a l'aplicació i, a més a més, que la contrasenya sigui la correcta 3.- L'usuari queda autenticat i identificat i pot accedir a l'aplicació

- **Excepcions**
    2. Si la identificació i autenticació donen resultat negatiu (l'usuari no existeix o no és qui diu ser) es finalitzarà immediatament el cas d'ús llençant una excepció per tal d'informar de la situació a la persona que intenta accedir a l'aplicació
- **Funció del sistema associada**: login.do

2.4.2 Alta d'un club

- **Cas d'ús**: Donar d'alta un club
- **Actors**
    1. Treballador d'un club (iniciador)
    2. Treballador de la Federació Catalana de Piragüisme
- **Propòsit**
  Federar un club per tal que pertanyi a la FCP i pugui gaudir dels seus serveis
- **Resum**
  Un club interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que l'identifiquen com a entitat esportiva (estatuts) i les dades que apareixen al formulari de sol·licitud de la FCP. Aquesta entitat enviarà un treballador del club a la FCP. Allí serà atès per un treballador de la FCP, el qual validarà i entrarà les dades. Un cop acabat, el club estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un club interessat en pertànyer a la FCP envia un dels seus treballadors a l'oficina de la institució catalana amb totes les dades necessàries per tal de realitzar la sol·licitud de la llicència. Aquestes dades són els estatuts (que l'identifiquen com a entitat esportiva) i les dades demanades en el formulari de sol·licitud) 2.- El treballador de la FCP validarà que l'entitat representada pel treballador del club és realment l'entitat que diu ser (aquesta tasca es realitzarà amb l'ajuda dels estatuts) 3.- El treballador del club farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...) 4.- El treballador de la FCP accedirà a la zona per donar d'alta als clubs.	
	5.- El sistema presentarà el formulari de sol·licitud de llicència per a un club.
6.- El treballador de la FCP omplirà el formulari per donar d'alta una entitat amb les dades que li proporcionarà el treballador del club. En aquest formulari també hi apareixerà el nom d'usuari i la contrasenya que permetran al club entrar a la zona restringida de la pàgina web de la federació per tal de poder fer posteriors consultes, renovar-se...	
	7.- El sistema emmagatzemarà les dades entrades i la llicència serà activa.

- **Excepcions**
    3. Si el treballador del club no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).
    7. Si el club ja existia, les dades no seran introduïdes altre cop, sinó que només es reactivarà la llicència.
- **Funció del sistema associada**: altaClub.do

- **Incís**
    1. La llicència d'un club és anual i sempre s'acaba el dia 31/12 de l'any en que s'ha sol·licitat.
    2. Les dades del formulari per donar d'alta a un club són les següents[4]:

        * Nom
        * Nif
        * DGE (coincidirà amb el Número de Llicència)
        * Nom d'usuari
        * Contrasenya
        * Data d'inscripció en el registre

        Adreça
        Localitat
        Codi Postal
        Província
        * Telèfon
        Fax
        Web
        Correu electrònic
        Número de membres
        * Tipus de llicència (estatal o autonòmica)

        Dades del directius (President, Vice-president, Secretaria i Tresoreria)
            * Nom
            * Primer Cognom
            Segon Cognom
            * Nif
            Telèfon

        Dades de les 5 possibles disciplines que pot impartir el club:
            Nom
            Número de jugadors
            Número de federats

        Dades dels 4 possibles tècnics que pot tenir el club
            Número de llicència
            Nom
            Primer Cognom
            Segon Cognom
            Càrrec

        Dades de les 3 possibles activitats que pot desenvolupar el club
            Nom
            Descripció

[4] Les dades que tinguin un * davant seran obligatòries.

2.4.3 Alta d'un esportista

En aquest cas, tindrem dos casos d'ús, ja que un esportista pot ser donat d'alta des de la FCP o des del seu club.

- **Cas d'ús**: Dona d'alta un esportista (des del punt de vista del club)
- **Actors**
    1. Esportista (iniciador)
    2. Treballador del club
- **Propòsit**
  Federar un esportista que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un esportista interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que el federi. El treballador del club serà l'encarregat de demanar la federació del federat a la FCP. Un cop acabat, el federat estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un esportista demana al seu club que el federi a la FCP (aquest pas pot ser substituït pel fet que sigui el propi club que li pregunti a l'esportista si es vol federar i aquest accepti), portant totes les dades necessàries per tal d'omplir el formulari de sol·licitud	
2.- El treballador del club validarà que l'esportista pertany realment a l'entitat (és a dir, mirarà si n'és soci)	
3.- El treballador del club accedirà a la zona per donar d'alta esportistes.	
	4.- El sistema presentarà el formulari de sol·licitud de llicència per a un esportista.
5.- El treballador del club omplirà el formulari per donar d'alta un esportista amb les dades que li haurà proporcionat el soci sol·licitant. No obstant, indicarà que no s'ha fet el pagament, ja que aquest s'ha de realitzar directament a la FCP	
	6.- Les dades seran introduïdes al sistema de manera que quedaran a l'espera de la seva validació (veure cas d'ús validar llicència d'un federat)

- **Excepcions**
  2. Si la validació dóna resultat negatiu (l'esportista no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.
  6. Si l'esportista ja havia existit com a federat, les dades no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.
- **Funció del sistema associada** clubAltaEsportista.do

- **Cas d'ús**: Donar d'alta un esportista (des del punt de vista de la FCP)
- **Actors**
    1. Esportista (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Federar un esportista que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un esportista interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que l'autoritzi amb el seu segell. L'esportista anirà a l'oficina de la FCP i demanarà ser federat. Un cop acabat, l'esportista estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un esportista que es vol federar recopila totes les dades necessàries per omplir el formulari i demana al seu club que l'autoritzi amb el seu segell.	
2.- El treballador del club validarà que l'esportista pertany realment a l'entitat (és a dir, mirarà si n'és soci) i li segellarà la sol·licitud en paper requerida per la FCP.	
3.- L'esportista es presentarà a l'oficina de la FCP i demanarà el seu ingrés com a federat.	
4.- El treballador de la FCP validarà que l'esportista pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
5.- L'esportista farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...)	
6.- El treballador de la FCP accedirà a la zona per donar d'alta a esportistes	
	7.- El sistema presentarà el formulari de sol·licitud de llicència per a un esportista.
8.- El treballador de la FCP omplirà el formulari per donar d'alta un esportista amb les dades que li haurà proporcionat el soci sol·licitant.	
	9.- Les dades seran introduïdes en el sistema i la llicència serà activa.
10.- El treballador del la FCP rebrà la sol·licitud en paper i procedirà a crear la llicència (farà el carnet de federat)	

- **Excepcions**
  2. Si la validació dóna resultat negatiu (l'esportista no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si l'esportista no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).
  8. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.

8 i 11. Si l'esportista ja havia existit com a federat, en el moment en que el treballador de la FCP introdueixi el seu número de llicència, el formulari es carregarà amb les dades ja existents a la base de dades i aquestes no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.

- **Funció del sistema associada** altaEsportista.do
- **Incís**
    1. La llicència d'un esportista pot ser d'un, dos o tres dies, mensual o anual.
    2. Les dades del formulari de sol·licitud d'alta seran les següents:

        * Nom
        Nif
        * Número de Llicència
        * Primer Cognom
        Segon Cognom
        * Data de naixement
        * Lloc de naixement
        * Adreça
        Localitat
        Codi Postal
        Província
        Telèfon
        Telèfon 2
        Correu electrònic
        * DGE del club del qual és soci
        Disciplina habitual
        Nacionalitat
        * Tipus d'afiliació (1, 2 o 3 dies, mensual o anual)
        Sexe
        Categoria
        * Tipus de llicència (estatal o autonòmica)

2.4.4 Alta d'un àrbitre

En aquest cas, tindrem dos casos d'ús, ja que un àrbitre pot ser donat d'alta des de la FCP o des del seu club.

- **Cas d'ús**: Dona d'alta un àrbitre (des del punt de vista del club)
- **Actors**
    1. Àrbitre (iniciador)
    2. Treballador del club
- **Propòsit**
  Federar un àrbitre que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un àrbitre interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que el federi. El treballador del club serà l'encarregat de demanar la federació de l'àrbitre a la FCP. Un cop acabat, l'àrbitre estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un àrbitre demana al seu club que el federi a la FCP (aquest pas pot ser substituït pel fet que sigui el propi club que li pregunti a l'àrbitre si es vol federar i aquest accepti), portant totes les dades necessàries per tal d'omplir el formulari de sol·licitud	
2.- El treballador del club validarà que l'àrbitre pertany realment a l'entitat (és a dir, mirarà si n'és soci)	
3.- El treballador del club accedirà a la zona per donar d'alta àrbitres.	
	4.- El sistema presentarà el formulari de sol·licitud de llicència per a un àrbitre.
5.- El treballador del club omplirà el formulari per donar d'alta un àrbitre amb les dades que li haurà proporcionat el soci sol·licitant. No obstant, indicarà que no s'ha fet el pagament, ja que aquest s'ha de realitzar directament a la FCP	
	6.- Les dades seran introduïdes al sistema de manera que quedaran a l'espera de la seva validació (veure cas d'ús validar llicència d'un federat)

- **Excepcions**
  2. Si la validació dóna resultat negatiu (l'àrbitre no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.
  6. Si l'àrbitre ja havia existit com a federat, les dades no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.
- **Funció del sistema associada** clubAltaArbitre.do

- **Cas d'ús**: Donar d'alta un àrbitre (des del punt de vista de la FCP)
- **Actors**
    1. Àrbitre (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Federar un àrbitre que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un àrbitre interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que l'autoritzi amb el seu segell. L'àrbitre anirà a l'oficina de la FCP i demanarà ser federat. Un cop acabat, l'àrbitre estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un àrbitre que es vol federar recopila totes les dades necessàries per omplir el formulari i demana al seu club que l'autoritzi amb el seu segell.	
2.- El treballador del club validarà que l'àrbitre pertany realment a l'entitat (és a dir, mirarà si n'és soci) i li segellarà la sol·licitud en paper requerida per la FCP.	
3.- L'àrbitre es presentarà a l'oficina de la FCP i demanarà el seu ingrés com a federat.	
4.- El treballador de la FCP validarà que l'àrbitre pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
5.- L'àrbitre farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...)	
6.- El treballador de la FCP accedirà a la zona per donar d'alta a esportistes	
	7.- El sistema presentarà el formulari de sol·licitud de llicència per a un àrbitre.
8.- El treballador de la FCP omplirà el formulari per donar d'alta un àrbitre amb les dades que li haurà proporcionat el soci sol·licitant.	
	9.- Les dades seran introduïdes en el sistema i la llicència serà activa.
10.- El treballador del la FCP rebrà la sol·licitud en paper i procedirà a crear la llicència (farà el carnet de federat)	

- **Excepcions**
  2. Si la validació dóna resultat negatiu (l'àrbitre no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si l'àrbitre no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).
  8. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.

8 i 11. Si l'àrbitre ja havia existit com a federat, en el moment en que el treballador de la FCP introdueixi el seu número de llicència, el formulari es carregarà amb les dades ja existents a la base de dades i aquestes no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.

- **Funció del sistema associada** altaArbitre.do
- **Incís**
    1. La llicència d'un àrbitre sempre és anual i s'exhaureix el dia 31/12 de l'any en que s'ha fet l'alta.
    2. Les dades del formulari de sol·licitud d'alta seran les següents:

        * Nom
        Nif
        * Número de Llicència
        * Primer Cognom
        Segon Cognom
        * Data de naixement
        * Lloc de naixement
        * Adreça
        Localitat
        Codi Postal
        Província
        Telèfon
        Telèfon 2
        Correu electrònic
        * DGE del club del qual és soci
        Nacionalitat
        Sexe
        Categoria AB (aigües braves)
        Categoria AT (aigües tranquil·les)
        Categoria CP (caiac polo)
        Categoria CM (caiac mar)
        Categoria EL (estil lliure)
        Categoria RF (ràfting)
        * Tipus de llicència (estatal o autonòmica)

        Les categories poden tenir els següents valors:
            Auxiliar
            Bàsic
            Nacional
            Internacional

2.4.5 Alta d'un tècnic

En aquest cas, tindrem dos casos d'ús, ja que un tècnic pot ser donat d'alta des de la FCP o des del seu club.

- **Cas d'ús**: Dona d'alta un tècnic (des del punt de vista del club)
- **Actors**
    1. Tècnic (iniciador)
    2. Treballador del club
- **Propòsit**
  Federar un tècnic que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un tècnic interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que el federi. El treballador del club serà l'encarregat de demanar la federació del tècnic a la FCP. Un cop acabat, el tècnic estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un tècnic demana al seu club que el federi a la FCP (aquest pas pot ser substituït pel fet que sigui el propi club que li pregunti a el tècnic si es vol federar i aquest accepti), portant totes les dades necessàries per tal d'omplir el formulari de sol·licitud	
2.- El treballador del club validarà que el tècnic pertany realment a l'entitat (és a dir, mirarà si n'és soci)	
3.- El treballador del club accedirà a la zona per donar d'alta tècnics.	
	4.- El sistema presentarà el formulari de sol·licitud de llicència per a un tècnic.
5.- El treballador del club omplirà el formulari per donar d'alta un tècnic amb les dades que li haurà proporcionat el soci sol·licitant. No obstant, indicarà que no s'ha fet el pagament, ja que aquest s'ha de realitzar directament a la FCP	
	6.- Les dades seran introduïdes al sistema de manera que quedaran a l'espera de la seva validació (veure cas d'ús validar llicència d'un federat)

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el tècnic no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.
  6. Si el tècnic ja havia existit com a federat, les dades no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.
- **Funció del sistema associada** clubAltaTecnic.do

- **Cas d'ús**: Donar d'alta un tècnic (des del punt de vista de la FCP)
- **Actors**
    1. Tècnic (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Federar un tècnic que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un tècnic interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que l'autoritzi amb el seu segell. El tècnic anirà a l'oficina de la FCP i demanarà ser federat. Un cop acabat, el tècnic estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un tècnic que es vol federar recopila totes les dades necessàries per omplir el formulari i demana al seu club que l'autoritzi amb el seu segell.	
2.- El treballador del club validarà que el tècnic pertany realment a l'entitat (és a dir, mirarà si n'és soci) i li segellarà la sol·licitud en paper requerida per la FCP.	
3.- El tècnic es presentarà a l'oficina de la FCP i demanarà el seu ingrés com a federat.	
4.- El treballador de la FCP validarà que el tècnic pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
5.- El tècnic farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...)	
6.- El treballador de la FCP accedirà a la zona per donar d'alta a esportistes	
	7.- El sistema presentarà el formulari de sol·licitud de llicència per a un tècnic.
8.- El treballador de la FCP omplirà el formulari per donar d'alta un tècnic amb les dades que li haurà proporcionat el soci sol·licitant.	
	9.- Les dades seran introduïdes en el sistema i la llicència serà activa.
10.- El treballador del la FCP rebrà la sol·licitud en paper i procedirà a crear la llicència (farà el carnet de federat)	

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el tècnic no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si el tècnic no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).
  8. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.

8 i 11. Si el tècnic ja havia existit com a federat, en el moment en que el treballador de la FCP introdueixi el seu número de llicència, el formulari es carregarà amb les dades ja existents a la base de dades i aquestes no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.

- **Funció del sistema associada** altaTecnic.do
- **Incís**

3. La llicència d'un tècnic sempre és anual i s'exhaureix el dia 31/12 de l'any en que s'ha fet l'alta.
4. Les dades del formulari de sol·licitud d'alta seran les següents:

* Nom
Nif
* Número de Llicència
* Primer Cognom
Segon Cognom
* Data de naixement
* Lloc de naixement
* Adreça
Localitat
Codi Postal
Província
Telèfon
Telèfon 2
Correu electrònic
* DGE del club del qual és soci
Nacionalitat
Sexe
Categoria AB (aigües braves)
Categoria AT (aigües tranquil·les)
Categoria CP (caiac polo)
Categoria CM (caiac mar)
Categoria EL (estil lliure)
Categoria RF (ràfting)
* Tipus de llicència (estatal o autonòmica)

Les categories poden tenir els següents valors:

Nivell 1
Nivell 2
Nivell 3
Iniciador
Monitor de piragüisme
Entrenador bàsic
Entrenador nacional

2.4.6 Alta d'un directiu

En aquest cas, tindrem dos casos d'ús, ja que un directiu pot ser donat d'alta des de la FCP o des del seu club.

- **Cas d'ús**: Dona d'alta un directiu (des del punt de vista del club)
- **Actors**
    1. Directiu (iniciador)
    2. Treballador del club
- **Propòsit**
  Federar un directiu que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un directiu interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que el federi. El treballador del club serà l'encarregat de demanar la federació del directiu a la FCP. Un cop acabat, el directiu estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un directiu demana al seu club que el federi a la FCP (aquest pas pot ser substituït pel fet que sigui el propi club que li pregunti a el directiu si es vol federar i aquest accepti), portant totes les dades necessàries per tal d'omplir el formulari de sol·licitud	
2.- El treballador del club validarà que el directiu pertany realment a l'entitat (és a dir, mirarà si n'és soci)	
3.- El treballador del club accedirà a la zona per donar d'alta directius.	
	4.- El sistema presentarà el formulari de sol·licitud de llicència per a un directiu.
5.- El treballador del club omplirà el formulari per donar d'alta un directiu amb les dades que li haurà proporcionat el soci sol·licitant. No obstant, indicarà que no s'ha fet el pagament, ja que aquest s'ha de realitzar directament a la FCP	
	6.- Les dades seran introduïdes al sistema de manera que quedaran a l'espera de la seva validació (veure cas d'ús validar llicència d'un federat)

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el directiu no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.
  6. Si el directiu ja havia existit com a federat, les dades no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.
- **Funció del sistema associada** clubAltaDirectiu.do

- **Cas d'ús**: Donar d'alta un directiu (des del punt de vista de la FCP)
- **Actors**
    1. Directiu (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Federar un directiu que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis de la llicència.
- **Resum**
  Un directiu interessat en pertànyer a la Federació Catalana de Piragüisme haurà recopilat totes les dades que apareixen al formulari de sol·licitud de la FCP i demanarà al seu club que l'autoritzi amb el seu segell. El directiu anirà a l'oficina de la FCP i demanarà ser federat. Un cop acabat, el directiu estarà federat.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un directiu que es vol federar recopila totes les dades necessàries per omplir el formulari i demana al seu club que l'autoritzi amb el seu segell.	
2.- El treballador del club validarà que el directiu pertany realment a l'entitat (és a dir, mirarà si n'és soci) i li segellarà la sol·licitud en paper requerida per la FCP.	
3.- El directiu es presentarà a l'oficina de la FCP i demanarà el seu ingrés com a federat.	
4.- El treballador de la FCP validarà que el directiu pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
5.- El directiu farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...)	
6.- El treballador de la FCP accedirà a la zona per donar d'alta a esportistes	
	7.- El sistema presentarà el formulari de sol·licitud de llicència per a un directiu.
8.- El treballador de la FCP omplirà el formulari per donar d'alta un directiu amb les dades que li haurà proporcionat el soci sol·licitant.	
	9.- Les dades seran introduïdes en el sistema i la llicència serà activa.
10.- El treballador del la FCP rebrà la sol·licitud en paper i procedirà a crear la llicència (farà el carnet de federat)	

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el directiu no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si el directiu no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).
  8. Si algun dels camps obligatoris del formulari no s'omplen el sistema traurà un missatge d'error indicant quin, o quins, camps manquen completar. No es podrà avançar en l'execució del cas d'ús fins que totes dels dades obligatòries tinguin un valor entrat per l'usuari.

8 i 11. Si el directiu ja havia existit com a federat, en el moment en que el treballador de la FCP introdueixi el seu número de llicència, el formulari es carregarà amb les dades ja existents a la base de dades i aquestes no seran introduïdes al sistema novament, sinó que només es renovarà la llicència.

- **Funció del sistema associada** altaDirectiu.do
- **Incís**
    5. La llicència d'un directiu sempre és anual i s'exhaureix el dia 31/12 de l'any en que s'ha fet l'alta.
    6. Les dades del formulari de sol·licitud d'alta seran les següents:

        * Nom
        Nif
        * Número de Llicència
        * Primer Cognom
        Segon Cognom
        * Data de naixement
        * Lloc de naixement
        * Adreça
        Localitat
        Codi Postal
        Província
        Telèfon
        Telèfon 2
        Correu electrònic
        * DGE del club del qual és soci
        Nacionalitat
        Sexe
        Categoria
        * Tipus de llicència (estatal o autonòmica)

2.4.7 Modificació i consulta de les dades d'un federat

En aquest cas, tindrem dos casos d'ús, ja que les dades d'un federat poden ser consultades i modificades des de la FCP o des del seu club.

- **Cas d'ús**: Modificació de les dades d'un federat (punt de vista del club)
- **Actors**
    1. Federat (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Canviar les dades d'un federat si aquestes han canviat durant la vigència de la llicència.
- **Resum**
  Un federat, les dades personals del qual han canviat, demana al club que les actualitzi. El treballador del club farà la modificació i, un cop aquesta sigui acceptada a la FCP, les dades quedaran modificades.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un federat demana al seu club que actualitzi les seves dades personals	
2.- El treballador del club validarà que el federat pertany realment a l'entitat (és a dir, mirarà si n'és soci o, per contra, és el propi club).	
3.- El treballador del club accedirà a la zona de consultar i modificar les dades correcta, segons si el federat és un àrbitre, un esportista, un tècnic, un directiu o el propi club.	
	5.- El sistema presentarà un mecanisme per cercar el federat.
6.- El treballador del club buscarà el federat i premerà l'opció de consulta	
	7.- El sistema presentarà un formulari amb les dades actuals de el federat.
8.- El treballador del club modificarà les dades modificables del federat que han canviat.	
	9.- El sistema emmagatzemarà les dades modificades com a no validades.
10.- El treballador del la FCP validarà la modificació de les dades (veure cas d'ús validació de dades d'un federat)	
	11.- El sistema emmagatzemarà les noves dades de forma permanent

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el federat no és soci del club) es finalitzarà immediatament el cas d'ús.
  10.- Si la validació ha resultat negativa les dades no es veuran modificades al sistema i el cas d'ús finalitzarà
- **Funcions del sistema associades**:
    1. modificacioClub.do: el federat és el propi club.
    2. modificacioEsportista.do: el federat és un esporitsta

3. modificacioArbitre.do: el federat és un àrbitre
4. modificacioTecnic.do: el federat és un tècnic
5. modificacioDirectiu.do: el federat és un directiu

- **Incís:**

  S'entenen com a dades no modificables les següents:

  Número de Llicència i Nif en el cas d'esportistes, àrbitres, tècnics i directius.

  Nom, Nif i Dge en el cas dels clubs

- **Cas d'ús**: Modificació de les dades d'un federat (punt de vista de la FCP)
- **Actors**
    1. Federat (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Canviar les dades d'un federat si aquestes han canviat durant la vigència de la llicència.
- **Resum**
  Un federat, les dades personals del qual han canviat, demana al club l'autoritzi a canviar les dades amb el seu segell. Un cop obtinguda l'autorització, el federat es presenta a la FCP i demana al seu treballador que li modifiqui les dades. El treballador de la FCP farà la modificació i les dades quedaran canviades.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un federat demana al seu club el segell per tal de poder modificar les seves dades	
2.- El treballador del club validarà que el federat pertany realment a l'entitat (és a dir, mirarà si n'és soci).	
3.- El treballador del club donarà el segell a el federat	
4.- El federat anirà a l'oficina de la FCP i demanarà una modificació de les seves dades.	
5.- El treballador de la FCP validarà que el federat pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
6.- El treballador de la FCP accedirà a la zona de modificar i consultar les dades corresponent, segons el federat sigui un club, un esportista, un àrbitre, un tècnic o un directiu.	
	7.- El sistema presentarà un mecanisme per cercar el federat.
8.- El treballador del club buscarà el federat i premerà l'opció de consulta	
	9.- El sistema presentarà un formulari amb les dades actuals de el federat.
10.- El treballador de la FCP modificarà les dades modificables de el federat que han canviat .	
	11.- El sistema emmagatzemarà les dades modificades de forma permanent

- **Cursos alternatius**
- **Excepcions**
  2. Si la validació dóna resultat negatiu (el federat no és soci del club) es finalitzarà immediatament el cas d'ús.
  5. Si la validació dóna resultat negatiu (el federat no porta el segell del seu club o el club no està federat) es finalitzarà immediatament el cas d'ús.
- **Funció del sistema associada**:
    1. modificacioClub.do: el federat és el propi club.
    2. modificacioEsportista.do: el federat és un esporitsta

3. modificacioArbitre.do: el federat és un àrbitre
4. modificacioTecnic.do: el federat és un tècnic
5. modificacioDirectiu.do: el federat és un directiu

- **Incís:**

  S'entenen com a dades no modificables les següents:

  Número de Llicència i Nif en el cas d'esportistes, àrbitres, tècnics i directius.

  Nom, Nif i Dge en el cas dels clubs

2.4.8 Renovació de la llicència d'un federat

En aquest cas, tindrem dos casos d'ús, ja que la renovació de la llicència d'un federat pot ser realitzada des de la FCP o des del seu club.

- **Cas d'ús**: Renovar federat (des del punt de vista del club)
- **Actors**
    1. Federat (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Renovar la llicència d'un federat que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis.
- **Resum**
  Un federat interessat en renovar la seva llicència demanarà al seu club que el federi. El treballador del club serà l'encarregat de demanar la renovació del federat a la FCP.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un federat demana al seu club que el renovi a la FCP (aquest pas pot ser substituït pel fet que sigui el propi club que li pregunti a el federat si es vol renovar i aquest accepti).	
2.- El treballador del club validarà que el federat pertany realment a l'entitat (és a dir, mirarà si n'és soci o és el mateix club qui es vol renovar).	
3.- El treballador del club accedirà a la zona per renovar corresponent, segons quin tipus de federat es vulgui renovar (el propi club, esportista, àrbitre, tècnic o directiu)	
	4.- El sistema presentarà un mecanisme per tal de cercar el federat
5.- El treballador del club cercarà el federat i el seleccionarà	
	6.- El sistema presentarà el formulari amb les dades de l'última llicència del federat
7.- El treballador del club premerà l'opció renovar.	
	8.- El sistema emmagatzemarà les dades com a pendents de validació de llicència (veure cas d'ús de validació de la llicència d'un federat)

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el federat no és soci del club) es finalitzarà immediatament el cas d'ús.
  4.- Si el treballador de la FCP no troba el federat significarà que aquest mai havia estat federat i, per tant, el cas d'ús finalitzarà immediatament.
  7.- Abans de renovar la llicència pot ser que el treballador del club hagi de canviar alguna de les dades personals del federat. Això implicarà que en el pas 8 no només s'hagi d'activar la llicència, sinó que les noves dades també quedaran emmagatzemades en el sistema.

- **Funcions del sistema associades**
    1. clubAltaClub.do: el federat és el propi club
    2. clubAltaEsportista.do: el federat és un esportista
    3. clubAltaArbitre.do: el federat és un àrbitre
    4. clubAltaTecnic.do: el federat és un tècnic
    5. clubAltaDirectiu.do: el federat és un directiu
- **Incís**
    1. Hi ha la possibilitat que el club decideixi renovar la sol·licitud d'un conjunt d'esportistes al mateix temps. En aquest cas, en el pas 4 hi haurà la possibilitat que el sistema cerqui federats no renovats i els presenti per pantalla. Un cop fet això, el treballador del club només haurà de seleccionar aquells que vulgui renovar (tenint en compte que, abans de renovar-los, hauran hagut de pagar la llicència).
    2. Si el que es pretén és renovar un conjunt d'esportistes, se n'haurà d'indicar el tipus d'afiliació (1, 2 o 3 dies, mensual o anual).

- **Cas d'ús**: Renovar un federat (des del punt de vista de la FCP)
- **Actors**
    1. Federat (iniciador)
    2. Treballador del club
    3. Treballador de la FCP
- **Propòsit**
  Renovar la llicència d'un federat que pertany a un club (aquest club ha de pertànyer a la FCP) per tal que pugui gaudir dels serveis.
- **Resum**
  Un federat interessat en renovar la seva llicència demana al club l'autoritzi a realitzar la renovació amb el seu segell. Un cop obtinguda l'autorització, el federat es presenta a la FCP i demana al seu treballador que li renovi la llicència. El treballador de la FCP farà la renovació.
- **Tipus**: Primària

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan un federat que vol renovar la seva llicència demana al seu club que l'autoritzi amb el seu segell.	
2.- El treballador del club validarà que el federat pertany realment a l'entitat (és a dir, mirarà si n'és soci) i li segellarà la sol·licitud en paper requerida per la FCP.	
3.- El federat es presentarà a l'oficina de la FCP i demanarà el seu reingrés com a federat.	
4.- El treballador de la FCP validarà que el federat pertany realment a un club, és a dir, demanarà el segell de l'entitat i haurà de comprovar que el club al qual pertany està federat.	
5.- El federat farà efectiu el pagament de la llicència (mitjançant un taló, diners en efectiu...)	
6.- El treballador de la FCP accedirà a la zona per renovar corresponent, segons si el federat és un club, esportista, àrbitre, tècnic o directiu	
	7.- El sistema presentarà un mecanisme per tal de cercar el federat
8.- El treballador de la FCP cercarà el federat i el seleccionarà	
	9.- El sistema presentarà el formulari amb les dades de l'última llicència de el federat
10.- El treballador de la FCP premerà l'opció renovar.	
	11.- El sistema reactivarà la llicència
12.- El treballador del la FCP rebrà la sol·licitud en paper i procedirà a crear la llicència (farà el carnet de federat)	

- **Excepcions**
  2. Si la validació dóna resultat negatiu (el federat no és soci del club) es finalitzarà immediatament el cas d'ús.
  4. Si la validació dóna resultat negatiu (el federat no duu el segell del seu club o el club al qual pertany no està federat) es finalitzarà immediatament el cas d'ús.
  5. Si el federat no fa efectiu el pagament, es finalitzarà immediatament el cas d'ús (les dades no seran introduïdes i la llicència no serà efectiva).

7.- Si el treballador de la FCP no troba el federat significarà que aquest mai havia estat federat i, per tant, el cas d'ús finalitzarà immediatament.

10.- Abans de renovar la llicència pot ser que el treballador de la FCP hagi de canviar alguna de les dades personals del federat. Això implicarà que en el pas 11 no només s'hagi d'activar la llicència, sinó que les noves dades també quedaran emmagatzemades en el sistema.

- **Funcinos del sistema associades**
    1. altaClub.do: el federat és un club
    2. altaEsportista.do: el federat és un esportista
    3. altaArbitre.do: el federat és un àrbitre
    4. altaTecnic.do: el federat és un tècnic
    5. altaDirectiu.do: el federat és un directiu
- **Incís**
    1. Hi ha la possibilitat que la FCP decideixi renovar la sol·licitud d'un conjunt de federats al mateix temps. En aquest cas, en el pas 7 hi haurà la possibilitat que el sistema cerqui els federats no renovats i els presenti per pantalla. Un cop fet això, el treballador de la FCP només haurà de seleccionar aquells que vulgui renovar (tenint en compte que, abans de renovar-los, hauran hagut de pagar la llicència).
    2. Si el que es pretén és renovar un conjunt d'esportistes, se n'haurà d'indicar el tipus d'afiliació (1, 2 o 3 dies, mensual o anual).

2.4.9 Validació de la llicència d'un federat

- **Cas d'ús**: Validació de la llicència d'un federat
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Analitzar les sol·licituds de federació d'un federat
- **Resum**
  El treballador de la FCP vol conèixer les noves sol·licituds de llicència per tal d'autoritzar-les
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant saber les sol·licituds d'alta que té pendents de validar	
2.- El treballador de la FCP accedirà a la zona de validació de llicències corresponent, segons si li interessa validar la llicència d'un club, esportista, àrbitre, tècnic o federat	
	3.- Presentarà totes les sol·licituds que estan pendents de validació amb un sistema que permetrà buscar-ne una de concreta.
4.- El treballador seleccionarà aquella sol·licitud que vulgui validar	
5.- El treballador de la FCP validarà la sol·licitud	
	6.- El sistema emmagatzemarà les dades de forma permanent i activarà la llicència

- **Excepcions**
- **Funcins del sistema associades**:
    1. ClubPermissionActivationAction.do: el federat és un club
    2. SportifPermissionActivationAction.do: el federat és un esportista
    3. ArbitratorPermissionActivationAction.do: el federat és un àrbitre
    4. TechnicianPermissionActivationAction.do: el federat és un tècnic
    5. DirectivePermissionActivationAction.do: el federat és un directiu

2.4.10 Validació de les dades personals d'un federat

- **Cas d'ús**: Validació de les dades personals d'un federat
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Analitzar les modificacions de dades que un club hagi pogut fer sobre ell mateix o sobre els seus federats
- **Resum**
  El treballador de la FCP vol conèixer les modificacions de dades fetes pels clubs sobre ells mateixos o sobre els seus federats
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant saber modificacions de dades que han realitzat els clubs	
3.- El treballador de la FCP accedirà a la zona de validació de dades corresponent, segons si li interessa veure les modificacions que un club ha realitzat sobre les seves pròpies dades o sobre esportistes, àrbitres, tècnics i directius que en siguin socis.	
	4.- Presentarà totes les modificacions realitzades per clubs pendents de validació
5.- El treballador seleccionarà aquella modificació que vulgui validar	
	6.- El sistema presentarà un formulari amb totes les dades del federat
8.- El treballador de la FCP validarà les dades	
	10.- El sistema emmagatzemarà les dades de forma permanent.

- **Excepcions**
  5.- Si la FCP no creu convenient realitzar la validació de dades, la modificació realitzada pel club quedarà denegada i s'esborrarà de la base de dades, deixant, per tant, la informació personal del federat tal i com estava abans de la modificació realitzada pel club.
- **Funcions del sistema associades**:
    1. notificacioClub.do: el federat és un club
    2. notificacioEsportista.do: el federat és un esportista
    3. notificacioArbitre.do: el federat és un àrbitre
    4. notificacioTecnic.do: el federat és un tècnic
    5. notificacióDirectiu.do: el federat és un directiu

2.4.11 Obtenció de les dades d'un federat per l'asseguradora

- **Cas d'ús**: Obtenció de les dades d'un federat per l'asseguradora
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Saber tots els moviments de renovació o donades d'alta que s'han efectuat durant el dia actual per tal de passar les dades a la mútua asseguradora.
- **Resum**
  El treballador de la FCP està interessat saber tots els moviments de renovació o donades d'alta que s'han efectuat durant el dia actual per tal de passar les dades a la mútua asseguradora
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant saber tots els moviments de renovació o donades d'alta que s'han efectuat durant el dia actual per tal de passar les dades a la mútua asseguradora	
2.- El treballador de la FCP accedirà a la zona de consultes mútua corresponent, segon si li interessa saber les dades per la mútua d'un esportista, àrbitre, tècnic o directiu.	
	3.- El sistema presentarà una taula amb totes les dades dels federats que s'han donat d'alta o renovat en el dia actual.
4- El treballador de la FCP premerà l'opció "obtenció versió imprimible"	
	5- El sistema crearà i descarregarà un arxiu excel per tal que es pugui enviar o imprimir.

- **Excepcions**
- **Funcions del sistema associades**:
    1. SportifInsurerSearchAction.do: el federat és un esportista
    2. ArbitratorInsurerSearchAction.do: el federat és un àrbitre
    3. TechnicianInsurerSearchAction.do: el federat és un tècnic
    4. DirectiveInsurerSearchAction.do: el federat és un directiu

2.4.12 Realització de cerques generals de federats

- **Cas d'ús**: Cerques Generals
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Realitzar consultes sobre els federats
- **Resum**
  El treballador de la FCP està interessat en dades (estadístiques) federats
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant en conèixer dades estadístiques federats	
2.- El treballador de la FCP accedirà a la zona de cerques generals	
	3.- El sistema presentarà un formulari amb 6 camps a omplir per tal de personalitzar la consulta (veure incís al final del cas d'ús)
4- El treballador de la FCP omplirà els camps en els que estigui interessat (1, 2..., o tots 6)	
5- El treballador de la FCP premerà l'opció de realitzar consulta	
	6- El sistema presentarà les dades obtingudes a partir de la consulta del treballador de la FCP
7 El treballador de la FCP premerà l'opció "obtenció versió imprimible"	
	8 El sistema crearà i descarregarà un arxiu excel per tal que es pugui enviar o imprimir.

- **Cursos alternatius**
- **Excepcions**
- **Funció del sistema associada**: GeneralSearchAction.do
- **Incís**
    1. Els camps pels quals serà possible realitzar una cerca són els següents:
        1.- Categoria
        2.- Sexe
        3.- Club
        4.- Disciplina habitual
        5.- Data d'alta
        6.- Any

    3 D'altra banda, es podran realitzar les consultes per:
    - Esportistes, àrbitres, tècnics i directius en conjunt
    - Esportistes
    - Àrbitres
    - Tècnics
    - Directius

2.4.13 Consulta de les categories d'esportistes existents

- **Cas d'ús**: Consulta de les categories
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Realitzar una consulta de les categories
- **Resum**
  El treballador de la FCP està interessat en conèixer el rang d'edats i anys que compren una categoria determinada
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant en conèixer les dades d'una determinada categoria esportiva 2.- El treballador de la FCP accedirà a la zona de categories 4- El treballador de la FCP farà la consulta desitjada.	3- El sistema presentarà una taula on, per cada categoria s'hi indicarà el rang d'edats i el rang d'anys de naixement que compren

- **Excepcions**
- **Funció del sistema associada**: CategoriaAction.do

2.4.14 Obtenció de la memòria amb les dades dels federats l'any anterior

- **Cas d'ús**: Obtenció de la memòria
- **Actors**
    1. Treballador de la FCP (iniciador)
- **Propòsit**
  Obtenció de la memòria amb les dades dels federats l'any anterior
- **Resum**
  El treballador de la FCP està interessat en conèixer les dades de tots els federats actius l'any anterior.
- **Tipus**

Accions realitzades pels actors	Respostes del sistema
1.- Aquest cas d'ús s'inicia quan el treballador de la FCP està interessant en conèixer les dades de tots els federats actius l'any anterior 2.- El treballador de la FCP accedirà a la zona de memòria 3- El treballador de la FCP indicarà l'any de la memòria que desitja obtenir 5- El treballador de la FCP descarregarà la memòria	4- El sistema buscarà la memòria i permetrà l'opció de descarregar-la

- **Excepcions**
    4.- Si la memòria de l'any demanat per l'usuari no existeix, el sistema informarà del problema a l'usuari i el cas d'ús finalitzarà immediatament.
- **Funció del sistema associada**: FindMemory.do

Capítol 3

Model de Domini

3.1 Introducció

Aquest breu capítol pretén reflectir i mostrar el model de domini que s'ha obtingut a partir dels requeriments pactats amb els usuaris per l'aplicació creada i explicada en aquestes pàgines.

3.2 Model de Domini

El model de domini és una representació visual de les classes conceptuals, o objectes, del món real en un àmbit de treball determinat. En el cas d'aquesta aplicació, l'àmbit de treball correspon a la Federació Catalana de Piragüisme [9].

La figura 3.2.1 conté el diagrama de classes que representa el model de domini de l'aplicació i a continuació es fa un breu descripció de les relacions que existeixen entre elles.

Classes que modelitzen els objectes del món real que intervenen en l'àmbit de la Gestió de Llicències de la FCP:

- Esportista: modelitza a un esportista del món real com a federat de la FCP.
- Tècnic: modelitza a un tècnic del món real com a federat de la FCP.
- Àrbitre: modelitza a un àrbitre del món real com a federat de la FCP.
- Directiu: modelitza a un directiu del món real com a federat de la FCP.
- Club: modelitza un club del món real com a entitat federada de la FCP.
- Llicència: modelitza una llicència de federat de la FCP.
- TècnicDelClub: modelitza un tècnic del club del món real
- Activitats: modelitza les activitats que el club realitza en el món real.
- Disciplines: modelitza les disciplines que el club imparteix en el món real.
- DirectiuDelClub: modelitza un directiu del club en el món real
- Disciplina: modelitza una disciplina que pot ser ensenyada per un tècnic o arbitrada per un àrbitre.

Pel que fa a les relacions podem comprovar que:

- Un federat només pot ser de quatre tipus: esportista, tècnic, àrbitre o directiu
- Té1: indica que un club té directius del club i tècnics del club, té unes disciplines que ensenya i un conjunt d'activitats que realitza[5].
- Soci: indica que un tècnic, àrbitre, directiu i esportista és soci d'un club.

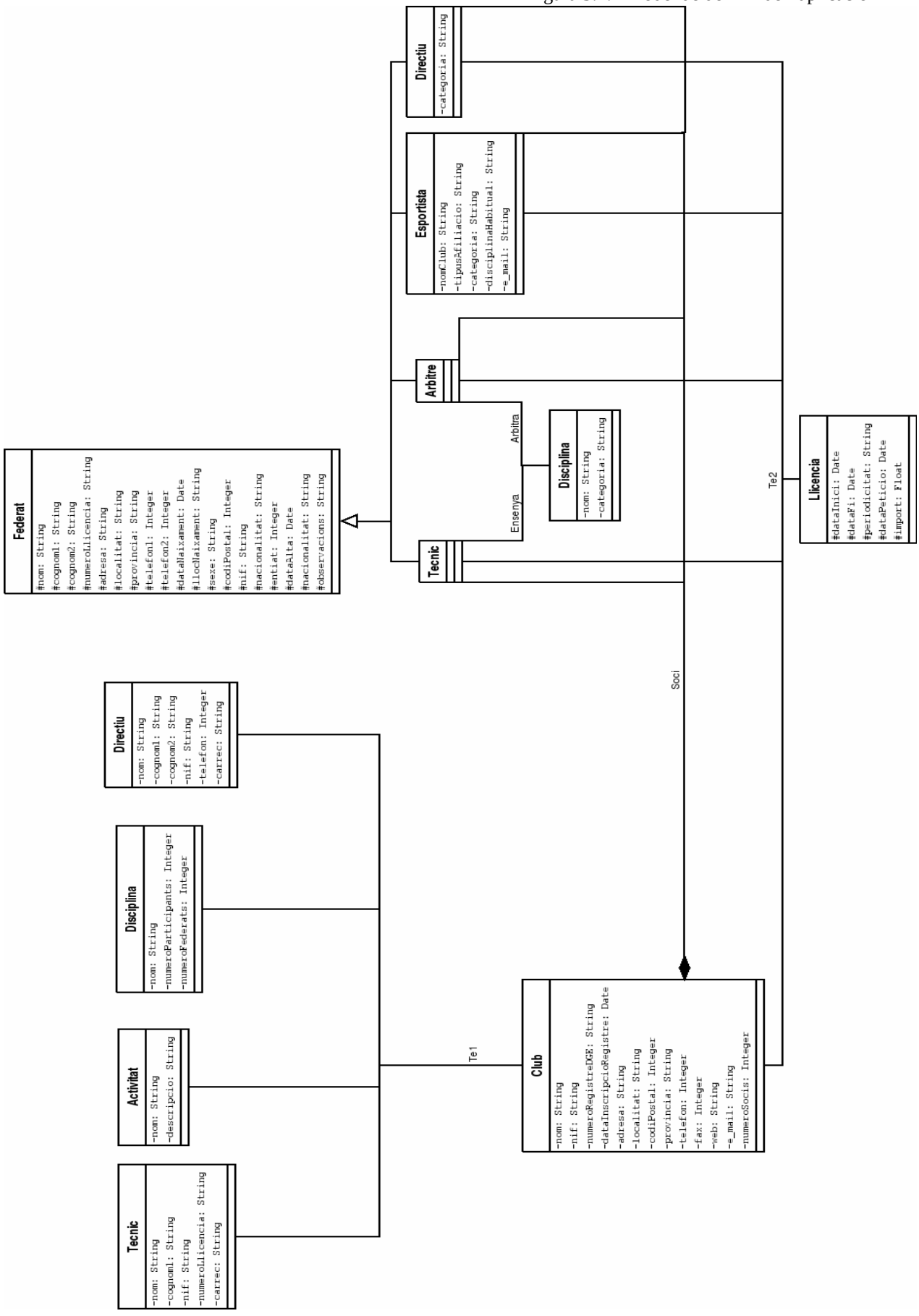
[5] Aquesta única relació en representa 4 de diferents (1 club té de 0 a 3 activitats; 1 club té de 0 a 4 tècnics del club, 1 club té 4 directius i 1 club té de 0 a 5 disciplines)) per tal de no carregar el diagrama de la figura 3.2.1 i fer-lo difícil d'entendre.

- Té2: indica el fet que un club, àrbitre, tècnic, esportista o directiu té una llicència, i una llicència només pot pertànyer a un únic club, un únic esportista, un únic àrbitre, un únic tècnic o un únic directiu[6].
- Ensenya: un tècnic pot ensenyar un conjunt de disciplines, per les quals tindrà una determinada categoria.
- Arbitra: un àrbitre pot arbitrar un conjunt de disciplines, per les quals tindrà una determinada categoria.

[6] Aquesta única relació en representa 5 de diferents (1 club té 1 llicència; 1 àrbitre té 1 llicència; 1 tècnic té 1 llicència; 1 esportista té 1 llicència i 1 directiu té 1 llicència) per tal de no carregar el diagrama de la figura 3.2.1 i fer-lo difícil d'entendre.

Model de Domini

figura 3.2.1 Model de domini de l'aplicació



Capítol 4

Aspectes de disseny i implementació

4.1 Introducció

En aquest capítol es pretén explicar amb tot detall com ha estat dissenyada l'Aplicació per a la Gestió de Llicències de la FCP i les eines utilitzades per tal d'implementar-la. Per tal d'aconseguir aquest objectiu es seguirà el patró següent:

1. En primer lloc s'explicarà l'arquitectura en tres capes que s'ha utilitzat per tal d'implementar aquest projecte.
2. Seguidament, es farà una descripció del patró "Model – Vista – Controlador" (MVC).
3. Posteriorment, es realitzarà l'explicació del disseny i implementació de la capa de domini o lògica.
4. Es realitzarà l'explicació de la capa de persistència.
5. Finalment, s'enunciaran i explicaran breument les tecnologies utilitzades .

4.2 Arquitectura en tres capes

L'arquitectura en tres capes permet dissenyar una aplicació amb tres blocs ben diferenciats i al mateix temps independents entre ells. Això implica dos grans avantatges:

a. Abstracció
b. Desacoblament

Gràcies a aquests dos punts a favor, es pot realitzar la implementació de cadascuna de les tres capes de manera completament independent.

Aquestes capes són:

1. Capa de presentació: és la capa més externa. La seva funció és comunicar-se amb l'usuari, rebre'n les peticions per tal de traspassar-les al sistema i presentar-li les dades. Podem dir que és la interfície de l'aplicació
2. Capa del domini, del negoci o lògica: és la capa intermitja. Conte tota la lògica que dóna les funcionalitats al sistema. S'encarrega de comunicar-se amb la capa presentació, per tal de rebre peticions i donar respostes, i amb la capa de persistència per tal de demanar dades.
3. Capa de persistència: és la capa més interna i la que conté emmagatzemades totes les dades. Està formada pel gestor de base de dades i rep peticions des de la capa domini per tal d'emmagatzemar, modificar o consultar informació.

En la figura 4.2.1 es pot veure un esquema d'aquestes tres capes aplicades a una aplicació web.

Per a obtenir més informació sobre l'arquitectura de tres capes vegeu [7] i [8]

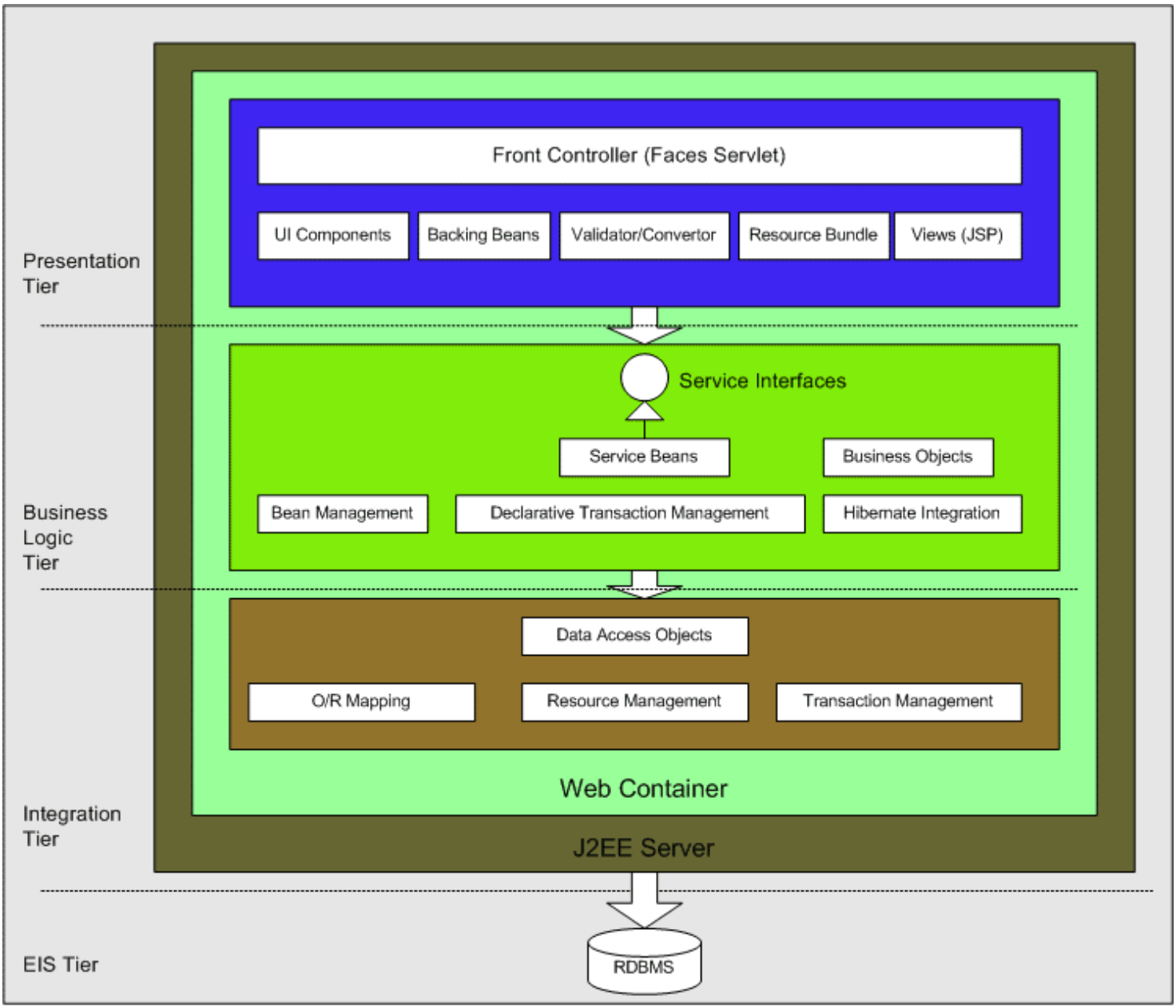


figura 4.2.1 Arquitectura de 3 capes en una aplicació web

4.3 Patró MVC

L'arquitectura en tres capes duu fàcilment a un disseny i implementació de l'aplicació seguint el patró "Model – Vista – Controlador", de tal manera que, tota aplicació dissenyada sota una arquitectura MVC consta de tres parts ben diferenciades:

1. Vista: interfície gràfica que permet comunicar l'usuari amb l'aplicació, presentar-li i demanar-li dades...
2. Model: es podria definir com el domini de l'aplicació, és a dir, totes les dades que poden ser tractades i les seves representacions.
3. Controlador: permet la comunicació i el flux d'informació entre la vista i el model.

Aquesta arquitectura proporciona els següents avantatges:

1. Desacoblament: la vista i el model no són dependents l'un de l'altre (per un mateix model, per exemple, podem tenir diferents vistes, és a dir, diferents maneres de presentar les dades a l'usuari)
2. Reutilització: el fet que la vista no estigui acoblada al model fa que aquest es pugui reutilitzar amb molta facilitat.

Finalment, dir que l'arquitectura MVC està edificada sobre altres patrons de disseny, com poden ser el patró Observador, Compost o Estratègia.

Per obtenir més informació sobre aquest patró i el seu ús en el treball realitzat, vegeu Annex B.

4.4 Capa de Domini

En el capítol anterior ja s'ha presentat el model de domini sorgit a partir dels requeriments establerts amb els usuaris. En aquest apartat, el que s'intentarà fer és una descripció del seu disseny i implementació seguint l'esquema següent:

a. Presentació de les JavaBeans que implementen el model de domini
b. Explicació del patró "Tempate Method" utilitzat en la seva implementació
c. Seguretat

4.4.1 Implementació del Model de Domini en JavaBeans

Com ja s'ha dit anteriorment, el model de domini de l'aplicació s'ha implementat mitjançant JavaBeans. S'ha creat una classe JavaBean per cada objecte del model de domini, de tal manera que tenim:

ActivitiesJB: implementa l'objecte Activitat
DisciplineJB: implementa l'objecte Disciplina
DirectiveJB: implementa l'objecte Directiu (del club)
ClubTechniciansJB: implementa l'objecte Tècnic (del club)
ArbitratorJB: implementa l'objecte Àrbitre
SportifJB: implementa l'objecte Esportista
TechnicianJB: implementa l'objecte Tècnic
CompleteDirectiveJB: implementa l'objecte Directiu
PremissionJB: implementa l'objecte Llicència

A més a més, a causa de la necessitat de presentar les dades a l'usuari en forma de taules s'ha creat la interfície TablePresentationJB que s'encarrega d'indicar el contingut de cada camp de la taula. Aquesta interfície és implementada per ClubJB, ArbitratorJB, SportifJB, TechnicianJB i CompleteDirectiveJB.

Finalment, també s'ha creat la classe ClubSetJB, la qual inclou en el seu interior un atribut ClubJB, un ActivitiesJB, un DisciplineJB, un DirectiveJB i un ClubTechniciansJB. D'aquesta manera aconseguim reunir en una sola classe totes les dades privades que ens interessen d'un club.

4.4.2 Patró "Template Method"

El patró "Template Method" s'encarrega de proporcionar una superclasse que conté un mètode que implementa l'esquelet d'un algoritme determinat, delegant a les subclasses la implementació d'alguns punts, o passos, d'aquest algorisme.

En aquest patró hi ha dos participants bàsics:

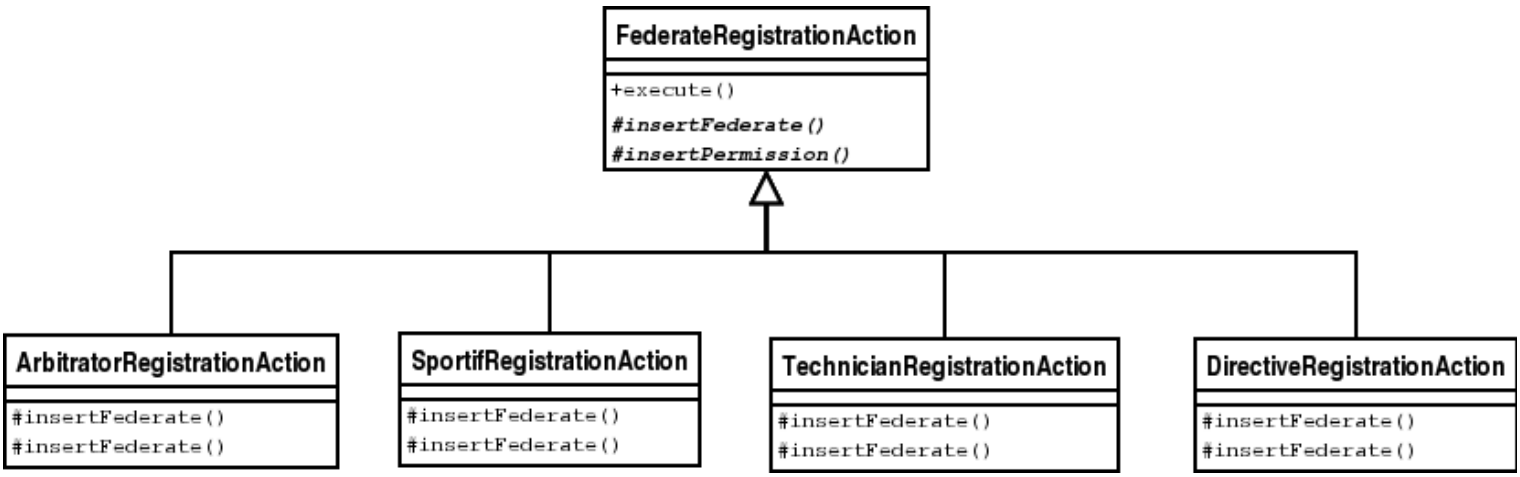
a. Classe Abstracta (superclasse): té dues tasques a fer:
    1. Definir el mètode plantilla que contindrà l'esquelet de l'algoritme.
    2. Definir un conjunt d'operacions primitives que hauran de ser implementades per les subclasses i que seran cridades des del mètode plantilla per tal de realitzar-ne certs passos.
b. Classe Concreta (subclasse): s'encarrega d'implementar les operacions primitives per tal de donar un comportament concret a l'algoritme estructurat per la superclasse.

L'ús de "Template Method" ha estat escaient en el projecte que tractem pel fet que, cadascuna de les funcionalitats implementades segueixen un patró comú canviant, tant sols, el tipus de federat sobre el qual operen.
Així per exemple, per donar d'alta un federat sempre haurem de fer els següents passos:

1. Mirar que s'hagi iniciat sessió
2. Comprovar que el club al qual pertany el federat té llicència activa
3. Inserir el federat
4. Inserir la llicència del federat

Tenint aquesta estructura clara, només s'ha de canviar el nom de federat pel d'esportista, àrbitre, tècnic o directiu. Així doncs, veient que l'estructura de donar d'alta és constant per qualsevol federat, s'ha extret de les classes concretes i s'ha dipositat en un mètode plantilla de la superclasse. Finalment, les subclasses només s'han hagut d'encarregar d'implementar el fet d'inserir el federat i la seva llicència per a un tipus concret.

La jerarquia de classes creada per aquest exemple concret quedaria de la següent manera:



Seguint aquesta metodologia, s'han implementat totes les funcionalitats del sistema.

Característiques que el patró "Template Method" aporta a l'aplicació:

a. Eliminació de la duplicació de codi: el fet de posar l'esquelet de la funcionalitat en una superclasse abstracta ens ha permès eliminar codi repetit en les subclasses que s'encarreguen de donar la implementació concreta.
b. Fàcil extensió: gràcies a la utilització de "Template Method" és molt fàcil incorporar nous tipus de federats al codi. Simplement hem de crear una subclasse per cadascun d'ells i fer que aquesta implementi els mètodes primitius de la superclasse, sense haver-se de preocupar de reinventar cada cop l'esquelet de la funcionalitat.
c. Reutilització de codi: l'esquelet de la funcionalitat podrà ser reutilitzat per totes les classes concretes que tractin amb cadascun dels federats.
d. Estructura de framework: gràcies a la utilització del patró "Template Method", cada funcionalitat del sistema té l'estructura d'un petit framework.

Per més informació sobre el patró "Template Method" vegeu [10] i [11].

4.4.3 Seguretat

Entenem per seguretat dos conceptes:

a. Cap usuari d'internet pot accedir a l'Aplicació per a la Gestió de Llicències de la FCP sense haver-se identificat i autenticat prèviament.
b. Un usuari del tipus Club no podrà accedir a les zones restringides (entenen com a zones restringides aquelles que atorguen funcionalitats úniques per als usuaris del tipus FCP).

Aquest dos conceptes permetran salvaguardar la informació de modificacions, o fins hi tot, eliminacions indegudes i, el que és més important, mantenir la privacitat de les dades personals i privades de tot federat o entitat.

Tot aquest sistema de seguretat s'ha implementat a la cada domini amb l'ajuda d'un atribut de sessió (logged), que indica si l'usuari que intenta accedir a l'aplicació s'ha identificat i autenticat correctament, i la classe AccessControl. Aquesta classe té la característica que segueix un patró "singleton", és a dir, només n'existirà una única instància i realitza el control d'accés d'usuaris a les zones restringides de l'aplicació.

4.5 Capa de persistència

Per tal de definir aquesta capa, cal parlar de tres conceptes clau:

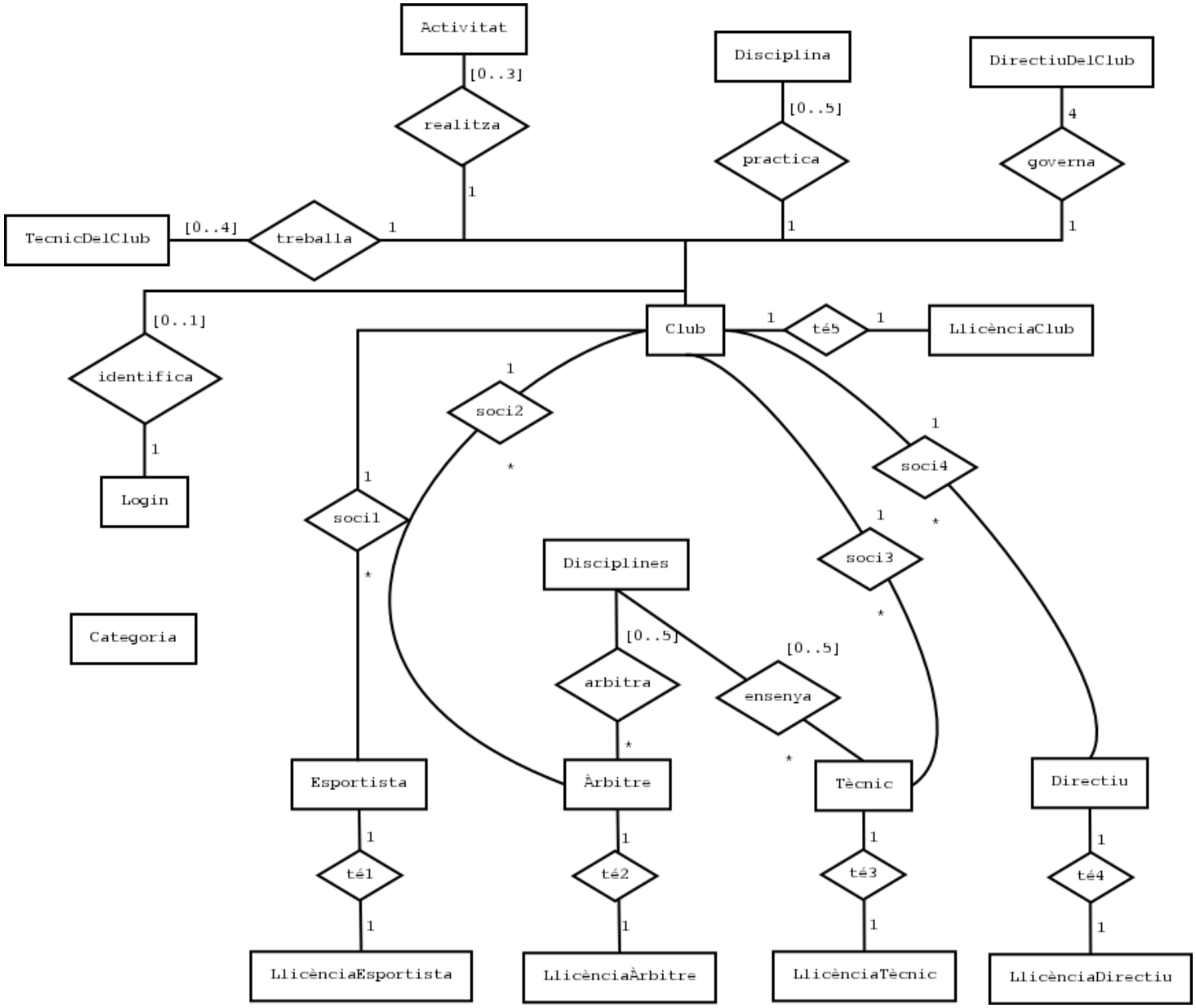
a. Model Entitat – Relació
b. Model Relacional
c. Classes embolcall de les taules de la base de dades

A continuació s'explicarà cadascun d'aquests elements en el marc del projecte que s'està documentant en aquestes pàgines.

4.5.1 Model Entitat - Relació

El model entitat - relació corresponent a la base de dades de l'aplicació es pot veure en la figura 4.5.1.1

figura 4.5.1.1 Model entitat – relació



4.5.2 Model Relacional

A continuació es realitzarà una descripció del Model Relacional creat a partir del Model Entitat – Relació de l'aplicació.

4.5.2.1 Explicació de les Entitats del Model Relacional creat

En el model relacional creat pel projecte que ens ocupa hi ha 18 entitats, o més ben dit taules de la base de dades, que intenten reflectir els objectes del món real que constitueixen la FCP en l'àmbit de la gestió de llicències. Aquestes entitats i els seus atributs són els següents[7]:

Login:
- *Nom d'usuari* (en cas que Tipus=Club, coincidirà amb el DGE del club al qual fa referància)
- Contrasenya
- Tipus (Club o FCP)

Club:
- *Número del registre DGE*
- Nom
- Nif
- Data d'inscripció al registre
- Adreça
- Localitat
- Codi Postal
- Província
- Telèfon
- Fax
- Pàgina web
- Correu electrònic
- Número de socis
- *Modificat*[8]

Activitat:
- *Nom*
- *DGE del club*[9]
- *Modificat*
- Descripció
- ClubModificat[10]

Disciplina:
- *Nom*

[7] Els atributs que es presenten en cursiva constitueixen la clau primària de l'entitat

[8] Tots els atributs "modificat" s'inclouen per tal de saber si les dades del federat han estat modificades pel club i, per tant, la FCP n'haurà de fer la validació per tal d'acceptar-les o denegar-les.

[9] Aquest atribut fa referència a l'atribut Número del Registre DGE de l'entitat Club i ens serveix per relacionar les activitats, disciplines, directius i tècnics amb el club al qual pertanyen.

[10] Aquest atribut fa referència a l'atribut modificat de l'entitat Club i indica si el club al qual pertany l'activitat, disciplina, directiu o tècnic ha estat modificat per ell mateix. ClubModificat i Modificat hauran de tenir el mateix valor.

Número de jugadors
Número de federats
DGE del club
Modificat
ClubModificat

DirectiuDelClub:
Nom
Primer Cognom
Segon Cognom
Telèfon
Nif
Càrrec
DGE del club
Modificat
ClubModificat

TècnicDelClub:
Nom
Primer Cognom
Segon Cognom
Nif
Número de llicència
Càrrec
DGE del club
Modificat
ClubModificat

LlicènciaClub
DGE del club (coincidirà amb el número de llicència)
Data d'alta
Data de finalització
Data de petició
Tipus (estatal o autonòmica)
Periodicitat (anual)
ClubModificat

Esportista:
Número de Llicència
Nom
Primer Cognom
Segon Cognom
Nif
Sexe
Adreça
Localitat
Província
Nacionalitat
Codi Postal
Telèfon
Telèfon auxiliar
Data de naixement
Lloc de naixement

DGE del Club
Categoria
Disciplina Habitual
Correu electrònic
Observacions
Modificat
ClubModificat

LlicènciaEsportista:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Esportista)
Data d'alta
Data de finalització
Data de petició
Tipus (estatal o autonòmica)
Periodicitat (1 dia, 2 dies, 3 dies, mensual o anual)
EsportistaModificat[11]

Àrbitre:

Número de Llicència
Nom
Primer Cognom
Segon Cognom
Nif
Sexe
Adreça
Localitat
Província
Nacionalitat
Codi Postal
Telèfon
Telèfon auxiliar
Data de naixement
Lloc de naixement
DGE del Club
Correu electrònic
Observacions
Modificat
ClubModificat

LlicènciaÀrbitre:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Àrbitre)
Data d'alta
Data de finalització
Data de petició
Tipus (estatal o autonòmica)
Periodicitat (anual)
ÀrbitreModificat[12]

[11] Aquest atribut fa referència a l'atribut "modificat" de l'entitat Esportista i serveix per indicar si l'esportista al qual pertany aquesta llicència ha estat modificat pel club del qual és soci. Per tal de mantenir la consistència de la base de dades, aquest atribut sempre haurà de ser fals.

DisciplinesÀrbitre:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Àrbitre)
Disciplina (AB, AT, CP, CM o RF)
Categoria (Auxiliar, Bàsic, Nacional o Internacional)
Modificat
ÀrbitreModificat[13]

Tècnic:

Número de Llicència
Nom
Primer Cognom
Segon Cognom
Nif
Sexe
Adreça
Localitat
Província
Nacionalitat
Codi Postal
Telèfon
Telèfon auxiliar
Data de naixement
Lloc de naixement
DGE del Club
Correu electrònic
Observacions
Modificat
ClubModificat

LlicènciaTècnic:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Tècnic)
Data d'alta
Data de finalització
Data de petició
Tipus (estatal o autonòmica)
Periodicitat (anual)
TècnicModificat[14]

DisciplinesTècnic:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Tècnic)
Disciplina (AB, AT, CP, CM o RF)

---

[12] Aquest atribut fa referència a l'atribut "modificat" de l'entitat Àrbitre i serveix per indicar si l'àrbitre al qual pertany aquesta llicència ha estat modificat pel club del qual és soci. Per tal de mantenir la consistència de la base de dades, aquest atribut sempre haurà de ser fals.

[13] Aquest atribut fa referència a l'atribut modificat de l'entitat Àrbitre i indica si l'Àrbitre al qual pertany la disciplina ha estat modificat pel club del qual és soci. ÀrbitreModificat i Modificat hauran de tenir el mateix valor.

[14] Aquest atribut fa referència a l'atribut "modificat" de l'entitat Tècnic i serveix per indicar si el tècnic al qual pertany aquesta llicència ha estat modificat pel club del qual és soci. Per tal de mantenir la consistència de la base de dades, aquest atribut sempre haurà de ser fals.

Categoria (Nivell 1, Nivell 2, Nivell 3, Iniciador, Monitor de piragüisme, Entrenador bàsic o Entrenador nacional)
Modificat
TècnicModificat[15]

Disciplines:[16]

Nom (coincidirà amb l'atribut, clau forana, "Disciplina" de les entitats, o taules, DisciplinesTècnic i DisciplinesÀrbitre).

Directiu:

Número de Llicència
Nom
Primer Cognom
Segon Cognom
Nif
Sexe
Adreça
Localitat
Província
Nacionalitat
Codi Postal
Telèfon
Telèfon auxiliar
Data de naixement
Lloc de naixement
DGE del Club
Categoria
Correu electrònic
Observacions
Modificat
ClubModificat

LlicènciaDirectiu:

Número de llicència (coincidirà amb l'atribut Número de Llicència de l'entitat Directiu)
Data d'alta
Data de finalització
Data de petició
Tipus (estatal o autonòmica)
Periodicitat (anual)
DirectiuModificat[17]

Categoria:

Nom
Edat d'inici
Edat de finalització

[15] Aquest atribut fa referència a l'atribut modificat de l'entitat Tècnic i indica si el tècnic al qual pertany la disciplina ha estat modificat pel club del qual és soci. TècnicModificat i Modificat hauran de tenir el mateix valor.

[16] Aquesta taula no s'ha inclós dins de la base de dades de l'aplicació.

[17] Aquest atribut fa referència a l'atribut "modificat" de l'entitat Directiu i serveix per indicar si el Directiu al qual pertany aquesta llicència ha estat modificat pel club del qual és soci. Per tal de mantenir la consistència de la base de dades, aquest atribut sempre haurà de ser fals.

4.5.2.2 Explicació de les Relacions del Model Relacional creat

En el model relacional creat pel projecte que ens ocupa hi ha 16 relacions que intenten reflectir la comunicació que existeix entre els objectes del món real que constitueixen la FCP en l'àmbit de la gestió de llicències. Aquestes relacions són els següents:

Té1: relació que ens indica la comunicació existent entre la taula Esportista i la taula LlicènciaEsportista. Un esportista té una única llicència.

Soci1: relació que ens indica la comunicació existent entre la taula Esportista i la taula Club. Un esportista és soci d'un únic club, mentre que un club pot tenir molts esportistes com a socis.

Té2: relació que ens indica la comunicació existent entre la taula Àrbitre i la taula LlicènciaÀrbitre. Un àrbitre té una única llicència.

Soci2: relació que ens indica la comunicació existent entre la taula Àrbitre i la taula Club. Un àrbitre és soci d'un únic club, mentre que un club pot tenir molts àrbitres com a socis.

Arbitra: relació que ens indica la comunicació existent entre la taula Àrbitre i la taula DisciplinesÀrbitre. Un àrbitre pot arbitrar de 0 a 5 disciplines.

Té3: relació que ens indica la comunicació existent entre la taula Tècnic i la taula Llicència Tècnic. Un tècnic té una única llicència.

Soci3: relació que ens indica la comunicació existent entre la taula Tècnic i la taula Club. Un tècnic és soci d'un únic club, mentre que un club pot tenir molts tècnics com a socis.

Ensenya: relació que ens indica la comunicació existent entre la taula Tècnic i la taula Disciplines Tècnic. Un tècnic pot ensenyar de 0 a 5 disciplines.

Té4: relació que ens indica la comunicació existent entre la taula Directiu i la taula Llicència Directiu. Un directiu té una única llicència.

Soci4: relació que ens indica la comunicació existent entre la taula Directiu i la taula Club. Un directiu és soci d'un únic club, mentre que un club pot tenir molts directius com a socis.

Realitza: relació que ens indica la comunicació existent entre la taula Club i la taula Activitat. Un club pot realitzar entres 0 i 3 activitats.

Practica: relació que ens indica la comunicació existent entre la taula Club i la taula Disciplina. Un club pot practicar entres 0 i 5 disciplines.

Governa: relació que ens indica la comunicació existent entre la taula Club i la taula DirectiuDelClub. Un club està governat per 4 directius (president/a, vice-president/a, secretari/a i tresorer/a).

Treballa: relació que ens indica la comunicació existent entre la taula Club i la taula TecnicDelClub. En un club poden treballar entre 0 i 4 tècnics.

Identifica: relació que ens indica la comunicació existent entre la taula Club i la taula Login. Un club pot realitzar s'identifica amb un únic login.

Té5: relació que ens indica la comunicació existent entre la taula Club i la taula LlicènciaClub. Un club té una única llicència.

Per tal d'implementar el model relacional descrit en aquests apartats, s'ha utilitzat el llenguatge SQL i el gestor de base de dades PostgreSQL.

4.5.3 Classes embolcall de les taules de la base de dades

En l'aplicació que ens ocupa, s'han creat el que anomenem DBBeans. Els DBBeans són classes encarregades de dur a terme la manipulació de la base de dades (per això les sigles DB, Data Base, del nom d'aquestes classes). Aquest tractament consistirà en:

1. Inserir dades
2. Modificar dades
3. Consultat dades
4. Eliminar dades

En l'aplicació per a la Gestió de Llicències de la FCP s'ha creat una DBBean per cada entitat del model relacional (a excepció de les entitats DisciplinesÀrbitre, el tractament de la qual s'inclou dins de la DBBean que tracta l'entitat Àrbitre, i DisciplinesTècnic, el tractament de la qual s'inclou dins de la DBBean que tracta l'entitat Tècnic).
A més a més, s'ha elaborat una DBBean específica que s'encarrega de les funcions d'obtenir i tancar una connexió amb la base de dades.

Així doncs, les classes creades són:

DBConnection: s'encarrega de l'obtenció i tancament de la connexió
DBClub: tracta amb l'entitat Club
DBClubPermission: tracta amb l'entitat LlicènciaClub
DBActivities: tracta amb l'entitat Activitat
DBDiscipline: tracta amb l'entitat Disciplina
DBClubTechnicians: tracta amb l'entitatTecnicsDelClub
DBDirective: tracta amb l'entitat DirectiuDelClub
DBArbitrator: tracta amb les entitats Àrbitre i DisciplinesÀrbitre
DBArbitratorPermission: tracta amb l'entitat LlicènciaÀrbitre
DBTechnician: tracta amb les entitats Tècnic i DisciplinesTècnic
DBTechnicianPermission: tracta amb l'entitat LlicènciaTècnic
DBSportif: tracta amb l'entitat Esportista
DBSportifPermission: tracta amb l'entitat LlicènciaEsportista
DBCompleteDirective: tracta amb l'entitat Directiu
DBDirectivePermission: tracta amb l'entitat LlicènciaDirectiu
DBLogin: tracta amb l'entitat Login
DBCathegory: tracta amb l'entitat Categoria

4.6 Tecnologies utilitzades

Per tal d'implementar l'Aplicació per a la Gestió de les Llicències de la FCP s'han utilitzat les tecnologies següents, organitzades segons la capa en que s'han explotat:

a. Capa de presentació
    1. JSP
    2. XHTML
    3. CSS
    4. Ajax i Javascript
b. Capa de domini
    1. Struts
    2. XML, DTD, XSL i DOM
c. Capa de persistència
    1. PostgreSQL

A més a més, s'ha utilitzat el contenidor Tomcat amb el "driver" jdbc3.

A continuació s'explicaran cadascuna d'aquestes tecnologies amb detall.

4.6.1 JSP

La tecnologia JSP permet crear pàgines web amb continguts tant estàtics com dinàmics. Els continguts estàtics poden presentar-se en formats tals com HTM, SVG i XML; mentre que els continguts dinàmics estan constituïts pels elements JSP.

Aquesta combinació d'elements estàtics amb capacitats dinàmiques és possible gràcies al fet que, qualsevol pàgina JSP es transformada pel servidor en una classe anomenada servlet". Aquesta transformació dóna com a resposta el contingut que veu l'usuari presentat en el navegador.

Els elements JSP bàsics que s'han utilitzat en la creació de les pàgines jsp del projecte són els següents:

- Directives: elements que especifiquen atributs de la pàgina i permeten importar llibreries d'etiquetes i paquets de codi java. La seva forma és la següent:

```
<%@ nom_directiva [atribut1=”valor1”...]%>
```

- Scriptlets: permeten incloure codi java a la pàgina jsp. La seva forma és la següent:

```
<%
    sentència1;
    ...
%>
```

- Expressions: permeten presentar per pantalla el valor que conté una variable java. La seva forma és la següent:

```
<%=variable%>
```

Per més informació sobre tecnologia JSP vegeu [14] i [15].

4.6.2 XHTML

4.6.2.1 Què és?

XHTML (Llenguatge de Marcat d'Hipertext Extensible) és una versió més estricta i neta d'HTML, que neix precisament amb l'objectiu de reemplaçar a HTML davant de la seva limitació d'ús amb les, cada vegada més abundants, eines basades en XML. XHTML amplia HTML 4.0 combinant la sintaxi d'HTML, dissenyat per mostrar dades, amb la d'XML, dissenyat per a descriure les dades.

4.6.2.2 Per a que serveix?

Davant de l'arribada al mercat d'un gran nombre de dispositius, XHTML sorgeix com el llenguatge, l'etiquetat del qual, més estricte que HTML, permetrà una correcta interpretació de la informació independentment del dispositiu des de el que s'accedeixi a ella. XHTML pot incloure tots els llenguatges com MathML, SMIL, o SVG, al contrari que HTML.

4.6.2.3 Exemples

A continuació es poden veure alguns exemples dels aspectes més importants a tenir en compte a l'hora d'utilitzar XHTML.

- Els documents han d'estar *ben formats*:
  Un format correcte en un document XHTML és molt important. Això vol dir que tots els elements han de tenir etiquetes de tancament, han d'estar escrits d'una forma determinada i, a més a més, tots els elements han d'estar aniuats correctament.
- Els noms d'atributs i elements han d'anar en minúscules:
  Tant els elements com els atributs han d'anar en minúscules per a tots els elements HTML i els noms dels atributs. Això és important ja que XML interpreta les majúscules i les minúscules de forma diferent.
  ```
  <body>Exemple correcte</body>
  <BODY>Exemple incorrecte</BODY>
  ```
- Els elements que no estiguin buits necessiten etiquetes de tancament:
- ```
  Exemple correcte.
  ```
- ```
  <p>Exemple correcte.</p>
  ```
  ```
  <p>Exemple incorrecte.<p>Exemple incorrecte.</p>
  ```
- Els valors de les etiquetes han d'anar sempre entre cometes:
  Tots els valors dels atributs han d'anar entre cometes, inclús aquells que siguin numèrics.
  ```
  <table rows="3">
  <table rows=3> exemple incorrecte
  ```

Per a una informació complerta sobre els canvis de HTML a XHTML, o per obtenir validadors, eines que canviïn d'un format a l'altre, i sobretot tota la informació referent a la codificació XHTML d'una pàgina web vegeu [12].

4.6.3 CSS

Les Fulles d'estil en cascada (*Cascading Style Sheets*, CSS) son un llenguatge formal utilitzat per a definir la presentació d'un document estructurat escrit en HTML o XML (i per extensió en XHTML). El W3C [23] és l'encarregat de formular l'especificació de les fulles d'estil que servirà d'estàndard per als navegadors.
L'idea que es troba darrera del desenvolupament de CSS es separar l'*estructu*ra d'un document de la seva *presentació.*
Per exemple, l'element d'HTML <*H1*> indica que un bloc de text és un encapçalament i que és més important que un bloc etiquetat com <*H2*>. Versions més antigues d'HTML permetien atributs extra dins de l'etiqueta oberta per donar-li format. No obstant cada etiqueta havia de tenir la informació, si es desitjava un disseny consistent.
Quan s'utilitza CSS, la etiqueta <*H1*> no ha d'aportar informació sobre com es visualitzarà, només marca l'estructura del document. L'encarregada de donar la informació sobre l'estil d'aquesta etiqueta serà una fulla d'estil on s'especifica com s'ha de mostrar <*H1*>. Aquesta informació de l'estil pot ser adjuntada tant en un document separat com en el mateix document HTML. En aquest últim es podrien definir estils generals a la capçalera del document o en cada etiqueta particular utilitzant l'atribut "*style"*.

Les avantatges d'utilitzar CSS son:

- Control centralitzat de la presentació d'un lloc web complert, amb el que s'agilita de forma considerable la actualització del mateix.
- Els navegadors permeten als usuaris especificar la seva pròpia fulla d'estils local que serà aplicada a un lloc web remot, amb el que augmenta considerablement l'accessibilitat. Per exemple, persones amb deficiències visuals poden configurar la seva pròpia fulla d'estils per augmentar la mesura del text o remarcar més els enllaços.
- Una pàgina pot tenir diferents fulles d'estil segons el dispositiu que la mostri o inclús a elecció de l'usuari.
- El document HTML és més clar d'entendre i s'aconsegueix reduir considerablement el seu tamany.

Per a afegir fulles d'estil a un document HTML només cal utilitzar la etiqueta <link> de la següent manera:

```
<link href="nuestra_hoja.css" rel="stylesheet" type="text/css" />
```

Se'n poden afegir més d'una però és important anar amb compte amb l'ordre en que es col·loquen, ja que alguns navegadors tenen problemes amb l'herència que porten implícita les CSS i podria ser que la visualització final del document no fos la desitjada.

Per més informació sobre CSS vegeu [13].

4.6.4 Ajax i javascript

La tecnologia Ajax i Javascript es troba profusament comentada i explicada en l'Annex A, per tant, en aquest apartat tant sols explicarem l'ús que se n'ha fet.

La tecnologia Ajax i Javascript ha estat utilitzada en aquest projecte per tal de realitzar consultes, tant dinàmiques com estàtiques, sobre la base de dades i presentar-ne els resultats sense necessitat de recarregar la pàgina al navegador.

S'entén com a consulta estàtica aquella que ha estat predefinida pels desenvolupadors per tal de satisfer una funcionalitat determinada del sistema[18].
S'entén com a consulta dinàmica aquella que depèn dels desitjos de l'usuari en un moment determinat[19].

Exemples de l'ús d'aquestes tecnologies els tenim a les pàgines:

- Categories: gràcies a Ajax i Javascript, es fa una petició al servidor per tal que vagi a buscar a la base de dades la informació sobre categories esportives, faci el còmput del rang d'edats i d'anys comprès en cada categoria i ho presenti tot per pantalla en forma de taula.
- Modificació i consulta: en aquesta pàgina, Ajax i javascript s'empren per realitzar dues tasques diferents:
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats.
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que compleixen amb els requeriments que l'usuari ha especificat en la zona de cerca que hi ha a la part superior de la pàgina.
- Validació de Llicència: en aquesta pàgina, Ajax i javascript s'empren per realitzar dues tasques diferents:
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tinguin validada la seva llicència.
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tenen validada la seva llicència i que, a més a més compleixen amb els requeriments que l'usuari ha especificat en la zona de cerca que hi ha a la part superior de la pàgina.
- Validació de Dades: en aquesta pàgina, Ajax i javascript s'empren per realitzar dues tasques diferents:
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tinguin validades les seves dades[20].
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tenen validades les seves dades i que, a més a més compleixen amb els requeriments que l'usuari ha especificat en la zona de cerca que hi ha a la part superior de la pàgina.

[18] Un possible exemple podria ser el següent: buscar tots els esportistes que, actualment, no tenen cap llicència activa
[19] Un possible exemple podria ser el següent: buscar tots els esportistes que actualment no tenen cap llicència activa i, a més a més, van néixer l'any 1967 i són de sexe femení.
[20] Les dades hauran estat modificades pel club del qual són socis i l'usuari de la FCP encara no les ha acceptat o denegat.

- Renovació: en aquesta pàgina, Ajax i javascript s'empren per realitzar dues tasques diferents:
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tinguin la llicència activa actualment.
    - S'envia una petició al servidor per tal que presenti, en forma de taula, tots els federats que no tenen llicència activa actuament i que, a més a més compleixen amb els requeriments que l'usuari ha especificat en la zona de cerca que hi ha a la part superior de la pàgina.
- Mútua: en aquesta pàgina, Ajax i javascript s'empren per enviar una petició al servidor per tal que presenti, en forma de taula, tots els federats que han estat donats d'alta el dia actual.
- Cerques Generals: en aquesta pàgina, Ajax i javascript s'empren per enviar una petició al servidor per tal que presenti, en forma de taula, tots els federats que compleixen amb els requeriments que l'usuari ha especificat en la zona de cerca que hi ha a la part superior de la pàgina.

4.6.5 Struts

El framework Struts es troba profusament comentat i explicat en l'Annex B, per tant, en aquest apartat tant sols explicarem breument l'ús que se n'ha fet.

El framework Struts ha estat utilitzat bàsicament per implementar l'Aplicació per a la Gestió de Llicències de la FCP seguin una arquitectura (o patró) MVC.

A més a més, també s'han aprofitat les seves capacitats d'internacionalització, validació de les dades i tractament de les excepcions.

4.6.6 XML, DTD, XSL i DOM

Primer de tot es realitzaran unes petites definicions d'aquest quatre elements:

- XML (Extensible Markup Language): és un meta-llenguatge que permet dissenyar multitud de llenguatges el tractament dels quals serà unificat. És una simplificació del SGML.
- DTD: document que serveix per pautar la gramàtica d'un llenguatge creat a partir d'xml, és a dir, en defineix l'estructura jeràrquica a nivell d'etiquetes, atributs i valor d'aquests. A més a més, també serveix per validar un document xml (un document xml serà vàlid sempre i quan compleixi amb l'estructura definida a la dtd amb el qual està associat).
- XSL: són les fulles d'estil d'XML, és a dir, l'equivalència de les CSS amb HTML. Permeten indicar com s'ha de presentar el contingut d'un fitxer xml.
- DOM: eina java que permet veure un document XML com un arbre. Aquesta visió permet la possibilitat de crear documents XML, parsejar-los, comprovar-ne la seva validesa i transformar-los a altres formats amb l'ajuda d'una DTD i d'un fitxer XSL.

Tota aquesta tecnologia explicada en aquest apartat ha estat utilitzada per tal de crear els fitxers en format xls que els usuaris desitjaven obtenir en 3 punts de l'aplicació:

1. A partir de les cerques generals realitzades sobre els federats.
2. A partir de les consultes per a l'asseguradora
3. Per tal d'obtenir la memòria amb les dades de tots els federats amb llicència activa de l'any anterior, o posteriors.

Per més informació, vegeu [16].

4.6.7 PostgreSQL

PostgreSQL és un gestor de bases de dades relacionals de codi lliure desenvolupat a la Universitat de Califòrnia de Berkeley.
Algunes de les característiques que ofereix són les següents:

- Herència
- Tipus de dades
- Funcions
- Constants
- Triggers
- Regles
- Integritat en les transaccions

Es basa en el llenguatge SQL i en el Model Relacional.

PostgreSQL ha estat utilitzat en aquest projecte com a gestor de base de dades.

Per més informació vegeu [17].

4.6.8 Tomcat

Tomcat és un contenidor de codi lliure desenvolupat sota la llicència "Apache Software License" [18].

Com a contenidor s'entén tota plataforma capaç d'implementar aplicacions i serveix web i de processar una petició per tal d'executar un determinat codi (en el nostre cas, codi java).

Tres conceptes a tenir en compte en el servidor Apache Tomcat són:

a. Variable d'entorn CATALINA_HOME: indicarà la ruta on es troba instal·lat el servidor.
b. Totes les aplicacions web es carreguen en el directori webapps en forma d'arxius war. Aquest arxius seran desplegats automàticament pel contenidor per tal que puguin ser accessibles des de la xarxa.
c. Tota aplicació web haurà de complir amb els següents requeriments:
    - Tindrà un directori WEB-INF on hi haurà el fitxer de configuració web.xml i els directoris classes (pels paquets de codi java desenvolupats per l'aplicació i compilats) i libs (pels arxius jar)
    - Tindrà un directori META-INF amb el fitxer de configuració context.xml.

La versió que ha estat utilitzada en aquest projecte ha estat la 5.5.12, la qual treballa sobre l'API de java versió 1.5. A més a més, s'ha combinat amb el driver jdbc3 per tal de realitzar el tractament de la base de dades creada amb el gestor PostgreSQL. Aquest driver s'ha instal·lat en els directoris $CATALINA_HOME/common/lib i $CATALINA_HOME/shared/lib.

Per més informació sobre Tomcat vegeu [18] i [19]

Capítol 5

Pressupost

5.1 Introducció

En aquest capítol es pretén fer un petit esbós del possible pressupost que podria tenir l'aplicació.

5.2 Pressupost

El fet que la intenció final de l'aplicació que s'està documentant és la d'entregar-la als usuaris de la federació, s'ha pensat en realitzar un pressupost més o menys adequat a la realitat per tal de mostrar una idea de com es valoraria la feina realitzada si s'hagués fet amb intenció de vendre-la.

El pressupost s'ha estructurat de la següent manera:
Llicències: tota aplicació utilitzada en el desenvolupament de l'aplicació ha tingut llicència GPL o de codi obert. D'igual manera, tota la documentació consultada estava llicenciada sota CC (Creative Commons) o era d'accés lliure. Així doncs, el cost de les llicències utilitzades ha estat de 0€.
Material:

Fungible:

Material d'escriptori ---> 12€
Desplaçaments --> 50€
Altres recursos --> 15€

Inventariable:

Ordenador portàtil --> 160€
Ordenador de sobre taula --> 110€

Hores*:* en aquest apartat es va suggerir d'utilitzar una funció que calcula la quantitat d'hores que presumptament s'havia invertit en el desenvolupament del projecte "COCOMO2" (vegeu [22]). No obstant, a partir d'una agenda electrònica que s'ha mantingut actualitzada durant el projecte s'ha comprovat que les hores reals no eren, ni s'aproximaven, a les previstes per l'algoritme.
El projecte ha durat un semestre, durant el qual hem comptabilitzat sobre unes 440hores de treball realitzades entre els dos membres del grup. Sobre el preu de l'hora s'han trobat una immensa quantitat de teories, i s'ha decidit optat per un preu dels més baixos, ja que els desenvolupadors són inexperts i han invertit gran part del temps en la investigació de les tecnologies emprades. S'ha decidit, per tant, que el preu de l'hora, sigui d'uns 10€/hora independentment del tipus de treball realitzat.
Així doncs: 440h * 10€/h = 4400€
En total el pressupost estaria sobre els 4900€ ja que s'hi ha afegit un marge de benefici per l'empresa i previsió de riscos en el pressupost.
A aquesta quantitat se l'hi ha de sumar el 16% d'I.V.A i s'obtindrà el preu total del software:

PRESUPOST: *5684€*

Capítol 6

Conclusions i Treball Futur

6.1 Conclusions

Després de tot el treball invertit en aquest projecte creiem que els objectius han estat assolits de manera satisfactòria, creant un codi fàcil d'estendre en cas d'incorporació de nous tipus de federats.

D'altra banda, des del punt de vista d'estudiants d'una enginyeria, creiem que el projecte de final de carrera ens ha servit, en certa manera, de síntesi, ja que ens ha permès aplicar molts coneixements adquirits durant els estudis. No obstant, no només ens hem quedat amb l'aplicació d'aquestes aptituds, sinó que també hem après a utilitzar i a treballar amb noves tecnologies, tals com el framework d'Struts i Ajax. A més a més, també ens ha servit per tenir un primer contacte senzill amb el món laboral que ens trobarem en un futur, ja que hem hagut d'aprendre a treballar amb i per als usuaris.

Finalment, ens agradaria comentar que tot i que estem molt contents amb tota l'aplicació realitzada en general, estem especialment satisfets amb el fet d'haver aconseguit un projecte via web que s'assimila, tant com és possible, a una aplicació d'escriptori en el sentit que les pàgines de la interfície no es recarreguen constantment gràcies a l'ajut de la tecnologia Ajax. I també volem fer especial menció al fet d'haver après a crear fitxers de format xls a partir de documents XML.

6.2 Funcionalitats, ampliacions i millores futures

Com ja es va citar en el capítol 2 d'Anàlisi de Requeriments, l'Aplicació per a la Gestió de Llicències de la FCP només satisfà una petita part dels desitjos dels usuaris. En aquest apartat explicarem cap a on s'hauria d'encaminar el desenvolupament de codi futur per tal de resoldre les mancances del projecte actual.

6.2.1 Funcionalitats futures

Les dues funcionalitats principals que s'haurien d'incloure al projecte són les demanades pels usuaris de la federació en la reunió del dia 09/08/2006. Aquestes funcionalitats addicionals són:

a. Gestionar la llicència d'empreses: implicaria implementar la donada d'alta, modificació i consulta de dades, validació de llicència, validació de dades i renovació de llicència per aquest nou tipus de federat o, més ben dit, entitat.
b. Creació del fitxer amb el format i la presentació de dades corresponent per tal de poder realitzar les impressions dels carnets de federat d'una manera més fàcil i simple.

A més a més, també es podria pensar en la gestió de bolcats de la base de dades que permetessin emmagatzemar la informació i, posteriorment, recuperar-la si fos necessari.

6.2.2 Ampliacions futures

La principal ampliació a que està subjecte aquest projecte és a realitzar una segona aplicació per a la gestió de competicions, tal i com es va dir en la reunió inicial amb els usuaris.

6.2.3 Millores futures

La principal millora que es creu que s'hauria de realitzar sobre el projecte és el desacoblament entre la capa presentació i la capa domini que hi ha actualment.

Per tal de realitzar la presentació en taules de totes les consultes realitzades s'ha utilitzat Ajax que es comunica amb el servidor per demanar-li la informació. Un cop el servidor rep la petició, la processa i crea el codi html necessari per crear les taules amb la informació rellevant dels usuaris.

El fet que en el codi java es creï directament codi html fa que el domini estigui lligat, acoblat, amb la presentació. El que es creu que s'hauria de fer és incloure la informació demanada per la presentació en un fitxer xml de tal manera que aquest es podria parsejar per tal d'obtenir-ne les dades i presentar-les a la interfície independentment de quina fos aquesta i amb què estigués implementada.

Annex A

Introducció a JavaScript

A.1 Conceptes bàsics

Els navegadors llegeixen documents HTML, per tant, les pàgines que trobem per la xarxa son documents HTML. No obstant, aquests documents son estàtics, immutables amb el pas del temps. Les pagines HTML es carreguen i aquí s'acaben els serveis de la pàgina. No ofereix ni dinamisme ni interacció amb el client. Per solucionar aquest problema neix JavaScript.

JavaScript *** és un llenguatge de tipus script compacte, basat en objectes y guiat per events, dissenyat específicament per al disseny d'aplicacions client-servidor dins l'àmbit d'internet.

A.2 Gramàtica del llenguatge

JavaScript és sensible a majúscules y minúscules, tots els elements de JavaScript han de referenciar-se com es varen definir prèviament (no es el mateix "Salt" que "salt").
El que es resumeix a la taula següent són els elements principals de la gramàtica de JavaScript.

Variables	Etiquetes que fan referència a un valor que pot canviar.
Operadors	S'utilitzen per calcular o comparar valors
Expressions	Qualsevol combinació de variables, operadors i declaracions que avaluen algun resultat
Sentències	Una sentència pot incloure qualsevol element de la gramàtica de JavaScript. Les sentències de JavaScript poden prendre la forma de condicional, bucle, o manipulacions del objecte. La forma correcta per a separar-les és per punt i coma (això només és obligatori si les declaracions múltiples resideixen a la mateixa línia. Tot i així és recomanable que s' acostumi a acabar cada instrucció amb un punt i coma, ja que estalvia problemes).
Objectes	Estructura emmagatzemadora de valors, procediments i funcions. Cada funció reflexa una propietat individual d'aquest objecte
Funcions	Una funció de JavaScript és bastant similar a un “procediment” o “subprograma” en altres llenguatges de programació. Una funció es un conjunt de sentències que realitzen alguna acció. Poden acceptar els valors entrants “paràmetres” i poden retornar un valor al finalitzar la funció.
Mètode	És una funció continguda dins d'un objecte.

Tot i que ja s'ha creat un estàndard sobre l'ús de la gramàtica de JavaScript, és aconsellable que abans de donar per finalitzada l'implementació de l'aplicació, es comprovi que aquesta funciona en els diferents navegadors.

A.3 Tipus de dades

Nombres	Enters o coma flotant.
Boleans	True o False.
Cadenes	Els tipus de dades cadena han d'anar delimitats per cometes simples o dobles.
Objectes	Obj = new Object();
Nuls	Null
Indefinits	Un valor indefinit és el que correspon a una variable que ha estat creada però no se li ha assignat cap valor.

A.4 Objectes bàsics del llenguatge

Els objectes propis del llenguatge són objectes que JavaScript proporciona per tal de facilitar el treball als programadors i proporcionar un entorn orientat a objectes amb tots el beneficis que això aporta. Els objectes més utilitzats son:

- String: Aquest objecte ens permet fer diverses manipulacions amb les cadenes. Quan assignem una cadena a una variable, JavaScript crea un objecte del tipus String per a fer les manipulacions corresponents amb els mètodes corresponents a aquest objecte.
- Array: Ens permet construir arrays, els elements dels quals poden ser qualsevol tipus bàsic. La longitud d'aquests arrays es modificarà dinàmicament sempre que afegim un nou element.
- Math: S'utilitza per realitzar càlculs en els scprits. Té la peculiaritat que les seves propietats no poden modificar-se, només consultar-se, ja que aquestes propietats són constants matemàtiques d'ús freqüent.
- Date: Ens permet interactuar amb dates, manipular-les, consultar-les i crear-ne de noves.
- Boolean: Ens permet crear booleans. No te mètodes.
- Number: Representar el tipus de dada “nombre”. Entre les seves propietats es troben els rangs de nombres que accepta.

A.5 Objectes del navegador

A.5.1 Jerarquia

És important, en el moment d'utilitzar JavaScript per fer modificacions de pàgines Web, tenir en compte la jerarquia d'objectes que es troba aniuada en una pàgina. En el esquema següent es mostren els objectes del navegador ordenats per l'ordre: "contenidor – contingut". A la dreta la directiva <HTML> associada a cada objecte:

```
* window
   + history
   + location
   + document          <BODY> ... </BODY>
      - anchor                <A NAME="..."> ... </A>
      - applet                <APPLET> ... </APPLET>
      - area                  <MAP> ... </MAP>
      - form           <FORM> ... </FORM>
         + button      <INPUT TYPE="button">
         + checkbox    <INPUT TYPE="checkbox">
         + fileUpload  <INPUT TYPE="file">
         + hidden      <INPUT TYPE="hidden">
         + password    <INPUT TYPE="password">
         + radio       <INPUT TYPE="radio">
         + reset       <INPUT TYPE="reset">
         + select      <SELECT> ... </SELECT>
            - options  <INPUT TYPE="option">
         + submit      <INPUT TYPE="submit">
         + text        <INPUT TYPE="text">
         + textarea    <TEXTAREA> ... </TEXTAREA>
      - image          <IMG SRC="...">
      - link                  <A HREF="..."> ... </A>
      - plugin                <EMBED SRC="...">
   + frame                    <FRAME>
 * navigator
```

A.5.2 Window

És l'objecte més elevat en la jerarquia del navegador ja que tots els elements d'una pàgina estan aniuats dins la finestra del navegador. Podria ser "navigator" també l'objecte més alt, però és un objecte independent en la jerarquia.

Les propietats més importants d'aquest objecte son:

- defaultStatus: cadena que conté el text per defecte que apareix a la barra d'estat del navegador.
- closed: és un booleà que ens diu si la finestra esta tancada o no.
- frames: és un array. Cada element d'aquest array (frames[0], frames[1], ...) és un dels frames que conté la finestra. El seu ordre s'assigna segons es defineixen en el document HTML.
- history: es tracta d'un array que representa les URLS visitades pel navegador.
- length: variable que ens indica quants frames te la finestra actual.
- location: cadena amb la URL de la barra de direcció.
- name: conté el nom de la finestra, o del frame actual.

- opener: és una referència al objecte window que l'ha obert, si la finestra ha estat oberta utilitzant el mètode open() d'aquest mateix objecte.
- parent: referència al objecte window que conté el frameset.
- self: és un nom alternatiu del window actual.
- status: String amb el contingut que te la barra d'estat.
- top: nom alternatiu de la finestra del nivell superior.

No es parlarà dels mètodes d'aquest objecte ja que no s'han utilitzat durant el projecte i tampoc són rellevants en aquesta breu documentació.

A.5.3 Frame

La finestra d'un navegador es pot dividir en diversos frames que continguin cadascun un document en el que mostrar continguts diferents. Tal i com passa amb les finestres, cadascun d'aquest frames pot ser referenciat, la qual cosa permet carregar documents en un marc sense afectar a la resta.

Cada frame es representa amb un objecte window, per tant, l'objecte frame té totes les propietats i mètodes del objecte window.

Alguns dels mètodes més utilitzats de frame i window son:
- alert(mensaje): mostra el missatge "mensaje" en un quadre de diàleg.
- confirm(mensaje): mostra un quadre de diàleg amb el missatge "mensaje" i dos botons, un d'acceptar i l'altre de cancel·lar. Retorna "true" si es prem acceptar i "false" si es prem cancel·lar.
- open(URL,nombre,caracteristicas): obra la URL que li passem com a primer paràmetre en una finestra de nom "nombre". Si aquesta finestra no existeix, s'obrirà una finestra nova en la que mostrarà el contingut amb les característiques especificades.

A.5.4 Location

Aquest objecte obté la URL actual així com algunes dades d'interès respecte aquesta adreça. La seva finalitat principal és permetre la modificació del objecte "Location" per a canviar a una nova URL, i extreure els components de la URL de forma separada per a poder treballar amb ells de forma individual. Una cadena URL és composa de la següent manera:
protocol://màquina_host[:port]/camí_al_recurs

Les propietats més importants d'aquest objecte son:
- hash: cadena que conté el nom de l'enllaç dins de la URL.
- host: cadena que conté el nom del servidor y el número del port dins de la URL.
- hostname: cadena que conté el nom de domini del servidor (o la direcció IP) dins de la URL.
- href: cadena que conté la URL completa.
- pathname: cadena que conté el camí al recurs,dins de la URL.
- port: cadena que conté el número de port del servidor dins de la URL.

- protocol: cadena que conté el protocol utilitzat (incloent els dos punts) dins de la URL.
- search: cadena que conté la informació passada en una crida a un script dins de la URL.

No es parlarà dels mètodes d'aquest objecte ja que tampoc s'han emprat en el projecte.

A.5.5 Document

El objecte document és el que engloba el contingut de tota la pàgina que s'està visualitzant. Això inclou el text, imatges, enllaços, formularis, ... Gràcies a aquest objecte es podrà afegir o modificar, dinàmicament, els continguts de la pàgina.

Les propietats més importants d'aquest objecte son:
- alinkColor: emmagatzema el color dels enllaços actius
- anchors: és un array amb els enllaços interns existents en el document
- applets: és un array amb els applets existents en el document
- bgColor: emmagatzema el color de fons del document
- cookie: és una cadena amb els valors de les cookies del document actual
- domain: guarda el nom del servidor que ha servit el document
- embeds: és un array amb tots els EMBED del document
- fgColor: emmagatzema el color de la primera plana del document
- forms: és un array amb tots els formularis del document. Els formularis contenen elements (caixes de text, botons, etc) que tenen les seves pròpies propietats i mètodes
- images: array amb totes les imatges del document
- lastModified: és una cadena amb la data de l'última modificació del document
- linkColor: emmagatzema el color dels enllaços
- links: és un array amb els enllaços externs
- location: cadena amb la URL del document actual
- referrer: cadena amb la URL del document que ha cridat a l'actual, en cas d'utilitzar un enllaç.
- Title: cadena amb el títol del document actual
- vlinkColor: emmagatzema el color dels enllaços visitats

Els mètodes més utilitzats són:
- clear(): neteja la finestra del document
- open(): obra l'escriptura sobre un document.
- close(): tanca l'escriptura sobre el document actual
- write(): escriu text en el document.
- writeln(): escriu text en el document, i a més a més, el finalitza amb un salt de línia.

A.6 AJAX

A.6.1 Introducció

El nom, AJAX, és un acrònim de *Asynchronous JAvaScrpit + XML*. Ajax no és una tecnologia pròpiament dita. Realment són diverses tecnologies que, tot i haver-se fiançat per mèrits propis, s'uneixen per crear una eina de treball nova i més potent. Les Tecnologies que AJAX incorpora son:

- Presentació basada en estàndards utilitzant XHTML y CSS.
- Mostra de resultats, dinàmicament gràcies al Document Object Model (DOM).
- Intercanvi i manipulació de dades utilitzant XML i XSLT.
- Recuperació asíncrona de dades mitjançant el objecte XMLHttpRequest.
- JavaScript per unir-ho tot.

El model clàssic de les aplicacions Web funciona d'aquesta manera: la gran majoria de les accions que l'usuari realitza sobre la interfície disparen una petició HTTP al servidor Web. Aquest realitza algun processament amb aquestes dades i retorna una pàgina HTML al client.

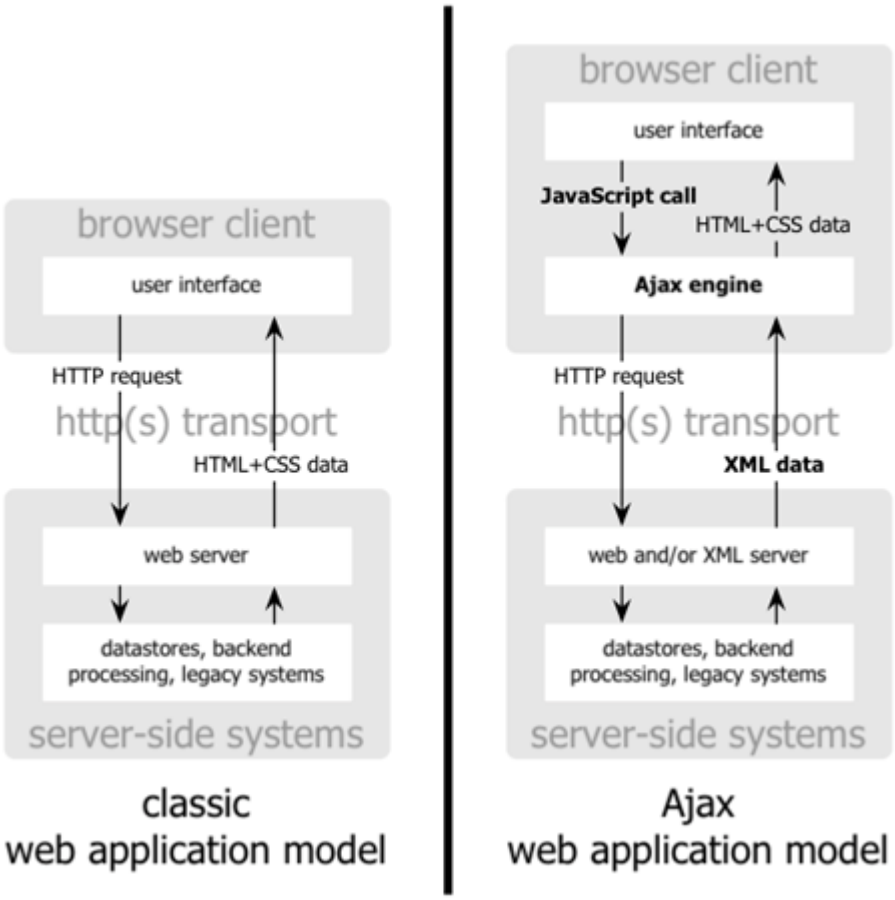


Figura 1: El model clàssic per a les aplicacions Web “esquerra” comparat amb el model de AJAX “dreta”

Aquesta aproximació té molt de sentit des d'un punt de vista tècnic, però no des del punt de vista de l'experiència de molts usuaris. El problema radica en el fet que, mentre el servidor esta processant la informació, l'usuari s'ha d'esperar i, per cada pas de l'acció total, l'usuari haurà d'esperar una mica més ja que el volum de dades augmentarà a cada volta.

Òbviament, si estiguéssim dissenyant la Web partint de zero i pensant en les aplicacions, no voldríem fer esperar als usuaris. Una vegada la interfície està carregada, per què la interacció del usuari s'ha d'aturar cada cop que l'aplicació necessita alguna cosa del servidor? De fet, per què l'usuari ha de veure sempre com l'aplicació va al servidor?
Donant resposta a aquestes dues preguntes, una aplicació AJAX elimina de la interacció amb la Web, la situació síncrona "d'arrancar-esperar-arrancar-esperar", introduint un intermediari, que anomenarem "Motor AJAX", entre l'usuari i el servidor. Podria semblar que afegir una capa més a l'aplicació la faria menys eficient, no obstant la realitat és que passa tot el contrari.

Al iniciar la sessió, en comptes de carregar una pàgina Web, el navegador carrega el motor AJAX, escrit en JavaScript i normalment emmagatzemat en un "frame" amagat. Aquest motor és el responsable de dos feines:

1. Mostrar la interfície que veu l'usuari
2. Comunicar-se amb el servidor en nom de la interfície.

El motor AJAX permet que la interacció de l'usuari amb l'aplicació sigui asíncrona (independent de la comunicació amb el servidor). Així l'usuari mai estarà mirant la finestra en blanc del navegador ni la icona del rellotge d'arena mentre espera que el servidor acabi.

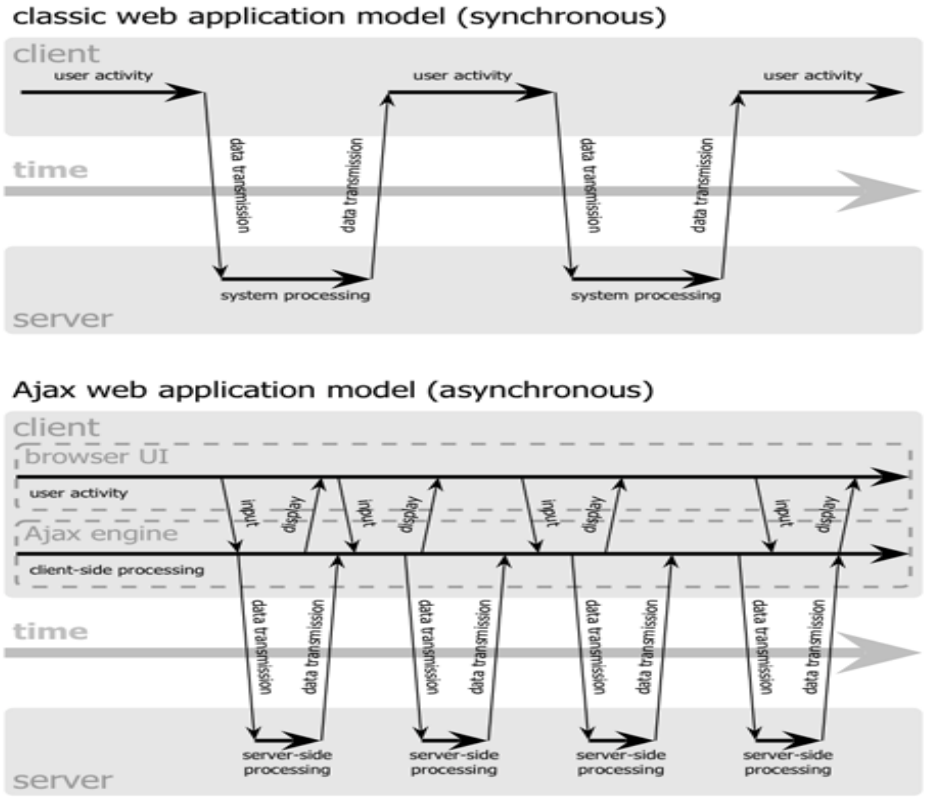


Figura 2: El patró d'interacció síncrona d'una aplicació Web clàssica comparada amb el patró asíncron d'una aplicació AJAX

Cada acció de l'usuari, que normalment generaria una petició HTTP, pren ara la forma d'una crida JavaScript al motor AJAX. Qualsevol resposta a una acció de l'usuari que no requereixi una operació del servidor (com una simple validació de dades, editar dades de memòria, fins i tot aspectes de navegació) serà manegat pel propi motor AJAX. Si aquest necessita ajuda del servidor per respondre (enviant dades per processar, carregant codi addicional, ...) realitza aquestes operacions de forma asíncrona, normalment utilitzant XML, sense frenar la interacció de l'usuari amb l'aplicació

A.6.2 Codificació Bàsica

Per a fer una breu introducció a com s'ha utilitzat la tecnologia AJAX en el projecte, es posarà un exemple del propi codi. En aquest cas la funció serverCall(url, id) servirà per afegir text a una secció de la pàgina que ens vindrà donada pel seu "identificador":

```
function serverCall (url, id)
{
	identificator=id;
	if (window.XMLHttpRequest) //Non-IE browsers
	{
		request=new XMLHttpRequest ();
		request.onreadystatechange=processStateChange;
		try
		{
			request.open ("GET", url, true);
		}
		catch (e)
		{
			alert (e);
		}
		request.send (null);
	}
	else if (window.ActiveXObject) //IE
	{
		request=new ActiveXObject ("Microsoft.XMLHTTP");
		if (request)
		{
			request.onreadystatechange=processStateChange;
			request.open ("GET", url, true);
			request.send ();
		}
	}
}
function processStateChange ()
{
	if (request.readyState==4) //Complete
	{
		if (request.status==200) // OK response
		{
			document.getElementById
			(identificator).innerHTML=request.responseText;
		}
		else
		{
			alert ("Error: " + request.statusText);
		}
	}
}
```

El funcionament és força trivial: la pàgina cridarà a la funció serverCall() com a resposta a un event de l'usuari. Aquesta funció crearà una instància de l'objecte XMLHttpRequest sobre el qual es passaran les dades de la pàgina web al servidor i, un cop hagin estat tractades i processades pel servidor, les dades tornaran a ser enviades a la pàgina per mitjà d'aquest mateix objecte. La funció encarregada d'obtenir les dades del servidor es processStateChange(), que espera a que el servidor hagi finalitzat correctament per inserir les dades a la pàgina utilitzant la propietat *"responseText"* de l'objecte XMLHttpRequest.

Gràcies a aquesta funció s'ha pogut realitzar cerques sobre la base de dades i mostrar els resultats a la mateixa pàgina, concretament dins del text encapsulat per la etiqueta amb el "id" que li s'ha passat a la funció serverCall(url,id). I tot això sense que l'usuari vegi com es recarrega la pàgina. L'ús realitzat d'AJAX potser no ha estat el més adient, ja que pels requeriments demanats per la Federació de Piragüisme no calia utilitzar la tecnologia AJAX.

A.6.3 XMLHttpRequest:

L'objecte XMLHttpRequest no és un estàndard del W3C. L'especificació de W3C DOM Level3 "load and save" conté algunes funcionalitats similars, però aquestes no estan implementades en cap navegador actualment. Per tant, si es necessita fer una petició HTTP des del navegador, haurem d'utilitzar l'objecte XMLHttpRequest. Aquest objecte està suportat a Internet explorer 5.0 o superior, safari 1.2, mozilla 1.0 / firefox, opera 9 i Netscape 7. Per tant, tot i no formar part de l'estàndard, està força acceptat pels navegadors més utilitzats.

Què és una petició HTTP? Amb una petició HTTP, una web pot fer una petició o obtenir la resposta d'un servidor, sense recarregar la pàgina. Per tant utilitzant l'objecte XMLHttpRequest es pot canviar les dades d'una pàgina web després que el servidor hagi carregat la pàgina.

De fet AJAX no és una tecnologia que s'hagi inventat ara, ja fa temps que s'utilitza aquest objecte per a fer pàgines dinàmiques, el que ha variat és l'èxit que ha tingut degut a l'ús molt ben encaminat que n'han fet Google i d'altres companyies. L'objecte XMLHttpRequest ja existia fa molt de temps i, com es pot veure, és la clau de la tecnologia AJAX ja que és qui ofereix les dues avantatges principals a AJAX :

- Peticions http
- Asíncronicitat

Com es crea un objecte XMLHttpRequest? Com ja hem explicat anteriorment, i per estrany que sembli, s'ha de tenir en compte el navegador per a poder crear un objecte XMLHttpRequest, per tant la sentencia de creació a Internet Explorer serà:
var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")
Mentre que a Mozilla, firefox, Opera, Safari i Netscape es crearà així:
var xmlhttp=new XMLHttpRequest()

Com s'ha fet prèviament amb els objectes del Navegador en l'apartat de Javascript, es realitzarà una petita introducció als mètodes i propietats d'aquest objecte per a veure quines possibilitats ens ofereix l'objecte i AJAX en general:

Mètode	**Descripció**
abort()	Cancel·la la petició actual
getAllResponseHeaders()	Retorna el conjunt de capçaleres http com un string
getResponseHeader("headername")	Retorna el valor de la capçalera especificada
open("method","URL",async,"uname","paswd")	Especifica el mètode, la URL i els altres atributs opcionals d'una petició. El paràmetre "method" pot prendre els valors "GET", "POST" o "PUT" El paràmetre "URL" ha de contenir la URL sobre la que es vol fer la petició El paràmetre async especifica si la petició ha de ser agafada de manera asíncrona o no.
send(content)	Envia la petició
setRequestHeader("label","value")	Afegeix una parella "label/value"a la capçalera http per a ser enviada.

Propietat	**Descripció**
onreadystatechange	És un manegador d'events que es llança cada canvi d'estat.
readyState	Retorna l'estat de l'objecte: 0= no inicialitzat 1=carregant 2=carregat 3= interactiu 4= completat
responseText	Retorna la resposta "response" com un string
responseXML	Retorna la resposta "response" com un document XML, aquest pot ser examinat i parsejat utilitzant els mètodes i propietats dels arbres W3C DOM.
status	Retorna l'estat com un nombre (Ex: 404 "NoT Found" o 200 "OK")
statusText	Retorna l'estat com un string (Ex: "OK")

A.6.4 Avantatges i inconvenients

Com totes les tecnologies, AJAX no és una solució per a tots els problemes, per tant és important saber en quins casos es convenient utilitzar-lo.
En primer lloc pararem atenció als avantatges que ofereix:

- Interactivitat: les aplicacions AJAX s'executen a la màquina del client, manipulant la pàgina actual dins del seu navegador utilitzant mètodes de DOM. Pot ser utilitzat per multitud de tasques com, per exemple, actualitzar o eliminar registres, expandir formularis, retornar peticions simples de ceca i/o editar àrbres de categories. Tot això sense tenir la necessitat de recarregar completament la pàgina HTML. Això permet desenvolupar aplicacions interactives amb interfícies d'usuari amb un coeficient de resposta més elevat gràcies a l'ús de les tècniques DHTML.
- Portabilitat: les aplicacions AJAX utilitzen característiques ben documentades, presents en tots els navegadors importants de la majoria de plataformes existents. Tot i que aquesta situació pot variar en el futur, actualment, els usos d'AJAX son efectius entre plataformes.

Tot i estar força acceptada, aquesta tecnologia també ha estat criticada. Les crítiques més “elaborades” fan referència a:

- Usabilitat: les aplicacions AJAX tenen tendència a acabar amb el comportament normal del botó “enrera” del navegador. Les diverses expectatives entre tornar a una pàgina modificada dinàmicament i tornar a una pàgina estàtica poden ser subtils. Els usuaris normalment esperen que fent click al botó “enrera” els portarà a l'última pàgina carregada, i en aplicacions AJAX el més probable és que això no passi. Per solucionar aquest problema s'ha observat que és possible seguir el comportament de l’usuari via callbacks que són cridats cada vegada que es prem el botó “enrera”, recuperant l'estat de l’aplicació que existia. Aquest problema d'usabilitat es detecta en el projecte que s’està documentat en aquestes pàgines en el moment de fer les cerques. No obstant, s'ha parlat amb els usuaris de l'aplicació i no s'ha tractat com un element prioritari a ser modificat. Es deixa com a possible actualització futura.
  Aquest mateix problema es detecta en el moment de guardar els marcadors/preferits en un moment particular de l'aplicació. Ja que el navegador guarda la URL i, tot i els esforços i les propostes de solucions a aquest problema, és poc probable que guardi la pàgina que l'usuari esta visualitzant en aquell mateix moment.
- Temps de resposta: l'interval entre la petició de l'usuari i la resposta del servidor s'ha de tenir en compte durant el desenvolupament d’AJAX. Si no es té en compte els usuaris poden experimentar esperes en la interfície de l'aplicació web i potser no s'ho esperen, o no ho entenen. Com a solució a aquestes esperes, s'acostuma a utilitzar un “feedback” visual per a informar a l'usuari de l'activitat que es realitza en segon pla.
- JavaScript: tot i que AJAX no requereix cap tipus de plug-in per al navegador, requereix que els usuaris tinguin JavaScript activat. Això s'aplica a tots els navegadors que suporten la tecnologia excepte per a Microsoft Internet Explorer 6 i anteriors, els quals necessiten tenir l’ActiveX activat, ja que el objecte XMLHttpRequest està implementat juntament amb el ActiveX en aquest

navegador. Aquest problema, no obstant, serà arreglat a la versió 7 del navegador de Microsoft.

Tal i com succeeix amb les aplicacions DHTML, les de AJAX també han de ser provades rigorosament per tal d'adaptar-se als diferents navegadors i plataformes. Tot i això, per internet podem trobar algunes llibreries per ajudar en aquestes feines de proves. A més a més, també s'han desenvolupat tècniques per assistir en el disseny d'aplicacions web que ofereixen alternatives per als usuaris que no tinguin JavaScript activat.

A.6.5 Conclusions

AJAX és una tecnologia desenvolupada sobre la Web, que pretén apropar les aplicacions Web a les aplicacions d'escriptori sense necessitat d'implementar applets o d'altres aplicacions que requereixin un plug-in especial al navegador.

Enfocant aquestes conclusions cap al projecte tractat, cal dir que no es tenia necessitat d'utilitzar AJAX ja que majoritàriament es realitzen manipulacions de la base de dades que no requereixen una excessiva interacció de l'usuari amb la pàgina. No obstant, ens vam proposar fer ús d'aquesta tecnologia a principis del projecte per a fer una interfície més agradable i que s'assembles a una aplicació d'escriptori tant com pogués ser. Aquest segon requisit el vam assumir després de veure que els usuaris de la Federació, que sempre han treballat amb una aplicació d'escriptori, ens van dir que "si era possible, procuréssim apropar la nostra aplicació a la seva per a facilitar el temps d'aprenentatge".

Annex B

Framework Struts

B.1 Introducció

Struts és un framework de codi lliure que facilita el desenvolupament d'aplicacions web en java basades en l'arquitectura "Model Vista Controlador (MVC)". En un primer moment, va ser ideat per Craig McClanahan i en l'actualitat forma part dels projectes de Jakarta [1].

Ofereix tres elements bàsics:

1. El "front controller" que s'encarrega d'assignar les tasques a les classes corresponents.
2. Selector de ruta, encarregat d'enviar la resposta a la petició
3. Un conjunt de llibreries d'etiquetes que permeten crear els formularis interactius.

El nucli del framework està creat a partir de tecnologies estàndards, com poden ser els servlets, JavaBeans, fitxers de propietats i XML, entre altres. A més a més, és compatible amb noves tecnologies, entre les quals es troba, per exemple, AJAX, també utilitzada en aquest projecte.

A continuació s'explicaran les diferents característiques del framework, basant els continguts en la versió 1.2.9, que ha estat la utilitzada en el treball (actualment, hi ha la versió 1.3.5 beta)

B.2 Arquitectura MVC

Tota aplicació dissenyada sota una arquitectura MVC consta de tres parts ben diferenciades:

4. Vista: interfície gràfica que permet comunicar l'usuari amb l'aplicació, presentar-li i demanar-li dades...
5. Model: es podria definir com el domini de l'aplicació, és a dir, totes les dades que poden ser tractades i les seves representacions.
6. Controlador: permet la comunicació i el flux d'informació entre la vista i el model.

Aquesta arquitectura proporciona els següents avantatges:

3. Desacoblament: la vista i el model no són dependents l'un de l'altre (per un mateix model, per exemple, podem tenir diferents vistes, és a dir, diferents maneres de presentar les dades a l'usuari)
4. Reutilització: el fet que la vista no estigui acoblada al model fa que aquest es pugui reutilitzar amb molta facilitat.

Finalment, dir que l'arquitectura MVC està edificada sobre altres patrons de disseny, com poden ser el patró Observador, Compost o Estratègia.

L'arquitectura MVC aplicada a un projecte web (en el cas que ens ocupa, una aplicació web en java) s'anomena patró "Front Controller", de tal manera que, els elements que el composen són:

1. Vista: la pàgina web (o codi html) que es presenta per pantalla (formularis, dades...)
2. Controlador: un servlet que s'encarregarà de distribuir les tasques entre un conjunt de classes cadascuna de les quals proporciona el codi per a una de les funcionalitats de l'aplicació.
3. Model: bases de dades i classes que s'encarreguen d'emmagatzemar informació (per exemple, els JavaBeans).

B.3 Avantatges i desavantatges d'utilitzar Struts per implementar l'arquitectura MVC enfront JSP

Avantatges:

- La configuració del funcionament del framework resta centralitzada en fitxers xml o de propietats. Aquest fet implica que molts canvis es podran dur a terme únicament canviant els valors definits en aquests arxius, estalviant el tràmit de retocar codi i tornar-lo a compilar posteriorment. A més a més, aquesta característica facilita la feina als desenvolupadors, ja que es poden centrar en la implementació pròpiament dita.
- A diferència de JSP, on els "FormBeans" s'havien de carregar utilitzant l'etiqueta <jsp:setProperty>, Struts ofereix la capacitat d'omplir aquest Beans automàticament, sense que el desenvolupador s'hagi de preocupar de res.
- Gràcies a les etiquetes ofertades en la seva llibreria "bean", Struts facilita la presentació de dades que, en JSP, es realitzava mitjançant l'ús de <jsp:useBean> i <jsp:getPropertu>
- Les etiquetes ofertades en la llibreria "html", les quals estan associades als "FormBeans", permeten presentar dades per defecte en un formulari o, fins i tot, tornar a presentar el mateix formulari amb les dades que s'havien introduït anteriorment.
- Struts permet una validació de les dades i tornar a presentar el formulari amb la finalitat d'informar a l'usuari dels errors comesos en la introducció de les mateixes.

Desavantatges:

- El fet d'aplicar l'arquitectura MVC mitjançant Struts en comptes de fer-ho amb JSP fa que el desenvolupador no només s'hagi de preocupar del disseny de la seva aplicació, sinó que també hagi de conèixer i entendre el funcionament de tot un framework de grans dimensions.
- La documentació existent sobre JSP és molt més extensa i clarificadora que la que fa referència a Struts.
- L'ús d'Struts fa que el funcionament de l'aplicació sigui menys transparent i, per tant, més difícil d'entendre i d'optimitzar.
- El fet que Struts estigui encarat a l'ús de l'arquitectura MVC fa que sigui difícil d'utilitzar en altres ambients o patrons.

B.4 Struts i l'arquitectura MVC

Con ja s'ha dit anteriorment, el framework Struts ofereix tota una estructura per proporcionar al desenvolupador un conjunt d'eines que l'ajudin a crear una aplicació web en java que segueixi una arquitectura MVC. Això implica que ha de garantir la possibilitat d'implementar els tres components bàsics d'aquesta arquitectura.
A continuació s'intenta explicar quina solució dóna Struts al "Model Vista Controlador"

B.4.1 Controlador

Dins de l'arquitectura MVC, el Controlador és l'element encarregat de comunicar l'usuari (les vistes) amb el domini de l'aplicació (el model). Aquest component centralitza totes les peticions de l'usuari, delega les tasques entre les diferents classes que té com a subordinades, les quals s'encarreguen d'operar sobre el model, i retorna una resposta.
En el framework Struts, el paper del controlador el juguen 5 elements específics:

- org.apache.struts.action.ActionServlet: classe o, més ben dit, servlet que rep les peticions de l'usuari i les delega al manegador de peticions per tal que aquest s'encarregui d'escollir quina funcionalitat s'ha de dur a terme. Es podria definir com el "Front Controller" de l'aplicació web.
- org.apache.struts.action.RequestProcessor: classe (manegador de peticions) que rep la petició de l'usuari que està tractant actualment "ActionServlet" i s'encarrega de trobar la classe que durà a terme la funcionalitat requerida. La cerca d'aquesta classe la realitzarà amb l'ajut del fitxer de configuració i de la classe org.apache.commons.digester.Digester (encarregada del parsejament del fitxer).
- org.apache.struts.action.Action[21]: superclasse que donarà la funcionalitat al sistema. Subclassificant-la podrem satisfer les necessitats de l'usuari. Cada subclasse que en creem serà mapejada a struts-config.xml i associada a una URL.
- ActionMapping i ActionForward[22]: classes que s'encarreguen de tractar el fitxer de configuració per tal de readreçar la petició a una "Action" o la resposta a una vista.
- struts-config.xml: fitxer de configuració que conté la informació necessària per associar una URL (una petició) a una classe concreta.

Clarificant idees, "ActionServlet" utilitza les classes "RequestProcessor" i Digester i el fitxer de configuració per trobar "l'Action" que realitza la funcionalitat demanada per l'usuari mitjançant la petició d'una URL, és a dir, "l'Action" que manipularà el model de manera adient. Un cop "l'Action" hagi acabat la seva tasca retornarà un "ActionForward" que contindrà la vista que s'ha de mostrar a l'usuari (més aviat contindrà un identificador que, buscant-lo al fitxer de configuració, ens donarà la vista a presentar).

[21] Hi ha documentació [1] en la qual s'inclou "Action" dins del Model, en comptes de considerar-la part del controlador. No obstant, després d'haver treballat amb el framework i haver-ne après el seu funcionament s'ha cregut convenient considerar "Action" com a element intrínsec del controlador, ja que és l'encarregada de donar la funcionalitat a l'aplicació (treballa sobre les dades, no és les dades). Aquesta idea de considerar "Action" com a part de la capa controlador bé suportada per altres fonts d'informació tals com [2].
[22] Ambdues classes pertanyents al paquet org.apache.struts.action

B.4.1.1 ActionServlet

La classe "ActionServlet", és la cara més externa de la capa controlador i es pot definir com el propi controlador (o front controller) de l'aplicació.

Les tasques que realitzarà, amb l'ajuda d'altres classes, són bàsicament 5 [2]:

1. Carregar els fitxers de configuració (mapejats com a <init-param> a web.xml en el cas d'utilitzar Tomcat).
2. Processar la petició de l'usuari.
3. Determinar quina tasca vol realitzar l'usuari segons la petició que ha enviat.
4. Tractar les dades del model.
5. Seleccionar la vista adequada a retornar al client.

Per cada petició que rebi de l'usuari, s'executarà el mètode "process (HttpServletRequest request, HttpServletResponse response)", el qual s'encarregarà de processar la petició i crear la vista corresponent. Per tal de fer-ho escollirà el "RequestProcessor" adient segons la petició i li passarà el control.

D'altra banda, la classe "ActionServlet" és, com el seu nom indica, un servlet i, com a tal, tindrà els mètodes init(), doGet (), doPost () i destroy (). A més a més, tindrà tot un conjunt d'operacions que li permetran dur a terme la càrrega dels fitxers de configuració.

Finalment, "ActionServlet" quedarà mapejat a l'arxiu de configuració del contenidor amb el qual treballem[23] de tal manera que se li passaran els paràmetres necessaris per al seu funcionament, en particular, la ruta on es troba el fitxer "struts-config.xml". Un exemple de mapeig (a web.xml) seria el següent:

```
<servlet>
        <servlet-name>controller</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
                <param-name>config</param-name>
                <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
</servlet>

<servlet-mapping>
        <servlet-name>controller</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

De tal manera que una URL de petició tindria aquest aspecte: ActionARealitzar.do (en cas de tractar-se d'un enllaç) o /ActionARealitzar (en cas de tractar-se d'un formulari).

[23] En el cas d'aquest projecte, el contenidor és Tomcat i el seu fitxer de configuració web.xml

B.4.1.2 RequestProcessor

La classe "RequestProcessor" centralitza la major part del processament de l'aplicació. Realitza numeroses tasques, les quals venen definides a continuació:

1. Processa la petició enviada per l'usuari per tal de trobar l'acció invocada. Aquesta feina la realitza en els seus mètodes processPath() i processMapping().
2. Crear "l'ActionForm" (o "FormBean, com també se'l coneix) associat amb la petició, l'omple amb les dades de la petició i, si és el cas, el valida. Tot això ho fa amb l'ajuda de tres mètodes: processActionForm(), processPopulate() i processValidate().
3. Crea "l'Action" a dur a terme i n'executa el seu mètode execute (). Tasques realitzades gràcies als mètodes processActionCreate() i processActionPerform().
4. Un cop finalitzada la feina realitzada per "l'Action" creada, es recull "l'ActionForward" que retorna per tal de readreçar el navegador a una vista. Això es fa dins del mètode processForwardConfig.

B.4.1.3 Action

La classe "Action" té com a finalitat realitzar les tasques demanades per l'usuari, és a dir, donar la funcionalitat a l'aplicació i retornar un "ActionForward" per tal de readreçar el control a un altre element del sistema (aquest element pot ser una vista, una altra "Action...). La consecució d'aquesta funcionalitat l'obtindrà gràcies al treball coordinat amb altres classes del sistema, tals com beans, classes d'accés a la base de dades..., les quals contindran la lògica apropiada.

D'altra banda, l'Action també es pot encarregar de realitzar comprovacions de seguretat com, per exemple, comprovar que l'usuari ha iniciat sessió i que no està intentant accedir a una zona restringida de l'aplicació.

Finalment indicar dues característiques:

1. El controlador únicament crea una instància "d'Action" per satisfer totes les peticions encarades a aquesta.
2. Tot el codi que s'hagi de dur a terme per aconseguir la meta desitjada s'ha de trobar dins del mètode execute()[24]. Així doncs, el desenvolupador s'haurà d'encarregar de subclassificar "Action" i sobreescriure'n aquest mètode per aconseguir realitzar les diferents tasques que pot dur a terme l'aplicació.

B.4.1.4 ActionMapping i ActionForward

Ambdues classes, "ActionMapping" i "ActionForward" treballen conjuntament amb una única finalitat: readreçar el control de l'aplicació a les diferents "Actions" o vistes, segons convingui, a partir de la petició feta per l'usuari i el resultat del processament d'aquesta.

[24] Una característica important d'aquest mètode és el fet que llença una única excepció en forma "d'Exception". Aquest fet atorga la capacitat de llençar qualsevol tipus d'excepció en el moment en que el desenvolupador el sobreescriu.

Per simplificar idees, podem fer la següent comparació o equivalència:

- "L'ActionMapping" seria la representació de la informació sobre el mapeig d'una determinada petició, que es troba en el fitxer de configuració "struts-confg.xml". El seu mètode més important és findForward(), el qual busca la informació de mapeig i retorna una ActionForward.
- "L'ActionForward" seria la representació del destí al qual se li ha d'atorgar el control de l'aplicació per tal de dur a terme una determinada tasca o presentar una vista.

Finalment, indicar que, per tal de realitzar el parsejament del fitxer de configuració, trobar la informació necessària i poder-la carregar a "l'ActionMapping" s'utilitza el Digester[25].

B.4.1.5 Fitxer de configuració struts-config.xml

Aquest fitxer conté el mapeig necessari per qualsevol petició que faci l'usuari, indicant "l'Action" a realitzar, "l'ActionForm" associat a la petició i la vista que es retronarà com a resposta.

Per tal que el controlador el pugui trobar i utilitzar, ha de quedar mapejat, com a paràmetre del servlet que actua de controlador ("ActionServlet"), en el fitxer de configuració del contenidor que s'utilitza. En el cas que ens ocupa, el contenidor és Tomcat i el seu fitxer de configuració web.xml, de tal manera que el mapeig quedaria de la manera següent:

```
<servlet>
        <servlet-name>controller</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
                <param-name>config</param-name>
                <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
</servlet>

<servlet-mapping>
        <servlet-name>controller</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

A continuació es realitzarà l'explicació de les etiquetes que constitueixen l'arrel i els tres mòduls (o elements) bàsics del fitxer.

<struts-config>
Arrel del fitxer xml

[25] El Digester pertany a Jakarta Commons i està al paquet org.apache.commons.digester.Digester.

<form-beans>

Dins d'aquesta secció es realitza el mapeig de totes les "ActionForm" (o FormBeans) existents en l'aplicació.

Per cada "ActionForm" que cal definir es crearà una etiqueta <form-bean>, la qual té dos atributs bàsics:

- "name": serà el nom de "l'ActionForm". S'utilitzarà per identificar la classe en els mapejos de les "Actions" que hi estiguin associades i per guardar-lo com atribut de "request" o "session".
- "type": nom complet (incloent el paquet on es troba) de "l'ActionForm" que s'ha d'invocar.

<action-mappings>

Dins d'aquesta secció es realitza el mapeig de totes les "Actions" creades pel desenvolupador, associant-les a una URL i configurant "l'ActionForm" que necessitaran i els retorns ("forwards" o traspàs de control) que podran provocar.

Per cada "Action" que cal definir es crearà una etiqueta <action>, la qual té 5 atributs bàsics:

- "path": URL que estarà associada al mapeig
- "'type": nom complet (incloent el paquet on es troba) de "l'Actio" que s'ha d'invocar.
- "name": nom de "l'ActionForm" que està associat a "l'Action". El valor d'aquest atribut correspondrà amb el de l'atribut "name" de l'etiqueta <form-bean> adequada.
- "scope": indicarà l'accessibilitat de "l'ActionForm" associada a "l'Action". Aquesta accessibilitat pot ser de "request" (només podrà ser accessible durant el tractament d'aquella petició), "session" (serà accessible mentre duri la sessió i des de qualsevol punt de l'aplicació) i "application" (serà accessible en tota l'aplicació, per totes les sessions).
- "input": element al qual es readreçarà el control si, en algun moment, l'execució de "l'Action" ha tingut un resultat inesperat i no tractat en el fitxer de configuració.

D'altra banda, dins de l'etiqueta <action> es pot afegir un altre tipus d'etiqueta: <forward>. Aquesta etiqueta indicarà cap a on s'ha de readreçar el control de l'aplicació un com l'execució de "l'Action" ha finalitzat retornant un determinat tipus "d'ActionForward". Aquesta etiqueta té dos atributs bàsics:

- "name": nom amb que "l'Action" identifica el readreçament de control per tal de crear "l'ActionForward" corresponent.
- "path": camí cap a on s'ha de readreçar el control. Pot ser l'adreça d'una pàgina jsp (en cas que, després de l'execució de "l'Action" en control ja passi a la vista) o un URI identificant una altra "Action" (en cas que la consecució d'una funcionalitat necessiti la coordinació entre diferents "Actions").

Finalment, dir que hi ha un altre ús d'aquesta secció en el qual es realitza el mapeig de pàgines jsp, en comptes "d'Actions". Per tal de realitzar-ho, s'utilitzar aquesta secció amb aquesta finalitat només fa falta crear una etiqueta <action> amb els

seus paràmetres “path” (correspondrà amb la URL demanada per l’usuari) i “forward” (indicarà la ruta on es troba la pàgina jsp)[26].

<global-forwards>

Dins d’aquesta secció es realitza el mapeig de les respostes a retornar després de processar una petició.

Dins d’aquesta sessió s’inclouen tots els traspassos de control que es realitzen en finalitzar una “Action” i que són comuns en més d’un d’aquests elements de la capa controlador.

Per cada traspàs de control es crearà una etiqueta <forward> amb exactament les mateixes característiques que l’etiqueta <forward> que es troba dins d’<action> dins d’<action-mappings>.

En el cas del projecte que ens ocupa, aquesta secció no s’ha utilitzat.

B.4.1.5.1 Petit exemple de fitxer de configuració struts-config.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE struts-config PUBLIC -//Apache Software Foundation//DTD Struts
Configuration 1.2//EN http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
    <form-bean name="formulari1" type="paquet.formulari1"/>
  </form-beans>

  <action-mappings>
    <action path="/accio1" type="paquet.accio1" name="formulari1" scope="request"
input="/WEB-INF/web/pgs/forumulari1.jsp">
      <forward name="correcte" path="/WEB-
      INF/web/pgs/Confirmacions/confirmacioAccio1.jps"/>
    <action>

    <action path="/menu1" forward="/WEB-INF/web/pgs/formulari1.jsp" />
  </action-mappings>
</struts-config>
```

Explicació de l’exemple:

Quan es premi el botó del menú, la URL del qual sigui “menu1.do” es readreçarà la vista cap a la pàgina formulari1.jsp. Aquesta pàgina contindrà un formulari amb la URL “/accio1”[27]. Quan aquest formulari s’enviï al servidor:

1. Es crearà un “FormBean” del tipus “paquet.formulari1” (que és el que està associat amb la petició), el qual només serà accessible per aquella petició (“scope” té valor “request”).

[26] En el projecte que s’ha realitzat, s’ha optat per aquest ús de la secció <action-mappings> amb la finalitat de mapejar els readreçaments fets en el menú de l’aplicació.

[27] Recordar que les URL dels enllaços es fan utilitzant el .do, mentre que en els formularis s’utilitza la / sense el .do (en aquest cas, posem .do perquè ha estat la manera en que s’ha exemplificat el mapeig del controlador a web.xml)

2. Es crearà una instància de “l’Action” paquet.accio1
3. Quan “l’Action” acabi la seva execució:
    a. Si ha estat correcta, retornarà un “ActionForward” identificada pel nom “correcte” i, per tant, es readreçarà el control a la pàgina confirmacióAccio1.jsp
    b. Si ha fallat en algun moment, es readrecarà el control a la pàgina formulari1.

B.4.2 Model

Dins de l'arquitectura MVC, el Model és l'element encarregat de representar el domini de l'aplicació, és a dir, totes les dades que poden ser tractades.

En el framework Struts, el model pot estar creat per la coordinació amb altres tecnologies com poden ser JDBC o EJB.

En el cas del projecte que es presenta en aquesta documentació, s'ha escollit la combinació d'Struts amb JavaBeans i el que hem anomenat DBBeans[28].
A continuació es farà una explicació d'aquests dos elements.

B.4.2.1 JavaBeans

Classes senzilles que s'encarreguen d'emmagatzemar les dades que corresponen a una entitat del domini de l'aplicació[29].
Es caracteritzen per tenir un constructor buit i funcions "get()" i "set()" per cadascun dels seus atributs.

En el projecte tractat, s'ha creat un JavaBean per cada entitat del domini.

B.4.2.2 DBBeans

Classes que s'encarreguen de tots els accessos que es realitzen a la base de dades, ja sigui per inserir dades, eliminar-les, modificar-les o consultar-les[30].
Es caracteritzen per tenir operacions de construcció, operacions per acceptar una connexió a la base de dades i operacions per realitzar el tractament de les dades emmagatzemades en aquesta.

En el projecte tractat, s'ha creat un DBBean per cada entitat de la base de dades relacional existent, és a dir, per cada taula de la base de dades[31]. Cada DBBean creat treballarà coordinadament amb el JavaBean corresponent per tal de controlar el flux de dades des de memòria principal a memòria secundària i viceversa

[28] Classes que s'encarreguen de realitzar els accessos a la base de dades mitjançant JDBC.
[29] En la documentació consultada [2] es fa referència a aquests tipus de classes com Beans de l'Estat del Sistema (System State Beans)
[30] En la documentació consultada [2] es fa referència a aquests tipus de classes com a Beans Lògics (Business Logic Beans)
[31] Al mateix temps, cada taula (entitat) de la base de dades correspon a una entitat del domini de l'aplicació.

B.4.3 Vista

Dins de l'arquitectura MVC, la Vista és la interfície gràfica que permet a l'usuari comunicar-se amb el sistema. A més a més, també s'encarrega de presentar-li les dades del domini de l'aplicació.

En el framework Struts, la vista pot estar creada per "JavaServer Pages" (pàgines jsp) utilitzant les llibreries d'etiquetes JSTL i les especificacions JSF [4].

En el cas del projecte que es presenta en aquesta documentació, s'ha escollit la combinació d'Struts amb pàgines jsp (utilitzant les facilitats de les llibreries d'etiquetes que proporciona el mateix framework) i els "FormBean".

A continuació es farà una explicació d'aquests dos elements.

B.4.3.1 Pàgines JSP i llibreries d'etiquetes

Les pàgines jsp són utilitzades en Struts per tal de crear la interfície gràfica que es presentarà a l'usuari i que reflexarà l'estat i les dades del domini. No obstant, les llibreries d'etiquetes tradicionals proporcionades per la tecnologia JSP seran substituïdes, o complementades, per les que ofereix el framework, la qual cosa simplificarà molt el moviment de les dades entre Model i Vista, passant pel Controlador.
Les llibreries d'etiquetes que ofereix el framework Struts són 5:

1. Llibreria d'etiquetes Html (struts-html-tld)
2. Llibreria d'etiquetes Bean (struts-bean.tld)
3. Llibreria d'etiquetes Lògiques (struts-logic.tld)
4. Llibreria d'etiquetes Aniuades (struts-nested.tld)
5. Llibreria d'etiquetes Estructurals (struts-titles.tld)

A continuació es farà una breu descripció de cadascuna d'elles donant-ne diferents exemples.

B.4.3.1.1 Llibreria d'etiquetes Html (struts-html.tld)

Aquesta llibreria proporciona totes les etiquetes necessàries per crear els formularis permetent, no només l'enviament de les dades introduïdes, sinó també la presentació del formulari amb dades per defecte (exemplificant-ne l'ús) o amb les dades introduïdes en un pas anterior per l'usuari[32].
Les etiquetes que es proporcionen en aquesta llibreria s'utilitzen en substitució de les etiquetes html tradicionals.

[32] Per exemple, ens podem troba el cas que hi hagi dades errònies; gràcies a aquesta capacitat, les dades podran tornar a ser presentades indicant els errors comesos sense que el desenvolupador s'hagi de preocupar d'implementar una manera de fer-ho.

Alguns exemples del contingut de la llibreria que s'han utilitzat en el projecte[33]:

- <html:html>: etiqueta que substitueix a la tradicional <html>
- <html:form>: etiqueta que substitueix a la tradicional <form> i que, a més a més, té com a mètode post com a mètode per defecte, en comptes de get.
- <html:text>: etiqueta que substitueix a la tradicional <input type"text">.
- <html:select>: etiqueta que substitueix a la tradicional <select>.
- <html:option>: etiqueta que substitueix a la tradicional <option>.
- <html:options>: etiqueta que substitueix a la tradicional <option> i que, a més a més, proporciona la capacitat de carregar un conjunt d'opcions al mateix temps
- <html:hidden>: etiqueta que substitueix a la tradicional <input type"hidden">.
- <html:submit>: etiqueta que substitueix a la tradicional <input type"submit">.
- <html:cancel>: etiqueta que substitueix a la tradicional <input type"cancel">.
- <html:errors>: etiqueta que permet mostrar un missatge d'error quan aquest es produeix (s'utilitza en la validació de dades).

De totes elles s'ha de dir que tenen un conjunt d'atributs, el més important dels quals és "property". Aquest paràmetre indica quin atribut del "ActioForm" (o "FormBean") associat al formulari s'haurà de presentar en aquesta etiqueta (o s'haurà d'omplir amb el contingut que l'usuari posi a l'etiqueta). Altres atributs interessants són:

- "styleId", equivalent a "l'id" tradicional.
- "styleClass", equivalent al "class" tradicional.
- Tots els atributs que permeten treballar amb javascript, tals com "onmonuseover", "onclick", "onfocus"...

B.4.3.1.2 Llibreria d'etiquetes Bean (struts-bean.tld)

Aquesta llibreria proporciona totes les etiquetes necessàries per crear i accedir a un "ActionForm" permetent la presentació de dades. A més a més, també ajuden al desenvolupador a internacionalitzar la interfície.
Les etiquetes que es proporcionen en aquesta llibreria s'utilitzen en substitució de les etiquetes jsp tradicionals.

Un parell d'exemples del contingut de la llibreria que s'han utilitzat en el projecte:

- <bean:message>: aquesta etiqueta permet internacionalitzar un missatge. El seu atribut "key" haurà de tenir el valor d'algun dels identificadors introduïts en un fitxer de propietats tal com MessageResources.properties.
- <bean:write>: permet presentar les dades d'un "ActionForm". Els dos paràmetres que ajuden a conseguir-ho són:
    - "name": serà el nom de "l'ActionForm" del qual se'n vol presentar una dada (aquest nom correspondrà amb el mapejat al fitxer de configuració struts-config.xml).
    - "property": nom de l'atribut de "l'ActionForm" el valor del qual es vol presentar per pantalla.

  Aquesta etiqueta substitueix a les tradicionals <jsp:useBean>i <jsp:getProperty>.

[33] Per tal d'utilitzar aquestes etiquetes se'ls hi ha de posar un prefix, indicant quin serà en la mateixa pàgina jsp. Per exemple: <%@ taglib url="(talgs/Struts-html" predix="html"%>

B.4.3.1.3 Llibreria d'etiquetes Lògiques (struts-logic.tld)

Aquesta llibreria proporciona etiquetes que permeten iterar sobre col·leccions d'objectes, generar condicionants i canviar el flux de la vista (fer readreçaments a altres pàgines). En altres paraules, ens permeten crear una lògica dins de la vista sense haver d'utilitzar codi java.

Alguns exemples del contingut de la llibreria:

- <login:redirect>: aquesta etiqueta permet readreçar el navegador a la pàgina indicada en el seu atribut "action" (normalment, el valor "d'action" no serà l'adreça directa de la pàgina jsp, sinó que s'haurà d'anar a buscar el mapeig a struts-config.xml). Aquesta etiqueta és molt útil en el cas que les pàgines jsp es vulguin posar dins del directori WEB-INF del Tomcat (contenidor utilitzat en aquest projecte), ja que una pàgina inicial ens readreçarà el navegador cap a l'interior del directori de manera transparent a l'usuari.
- <logic:forward>: tindria un funcionament similar a l'etiqueta explicada anteriorment.
- <logic:iterate>: aquesta etiqueta permet fer la iteració sobre una col·lecció d'objectes definida gràcies als atributs "name" i "property".
- <logic:empty>: aquesta etiqueta permet comprovar si el valor d'un determinat atribut, variable.. és nul. Això ho aconsegueix gràcies als seus paràmetres "name" i "property", els quals identifiquen l'objecte que es desitja comprovar si és nul.

B.4.3.1.4 Llibreria d'etiquetes Aniuades (struts-nested.tld)

Aquesta llibreria proporciona etiquetes que permeten relacionar (aniuar) etiquetes de les tres llibreries estàndard del framework (llibreria Html, Bean i Lògica). Així doncs, no proporcionen cap funcionalitat nova.

Per cada etiqueta de la llibreria Html, Bean o Lògica existeix l'equivalent en la llibreria Aniuada.

B.4.3.1.5 Llibreria d'etiquetes Estructurals (struts-titles.tld)

Aquesta llibreria proporciona etiquetes que permeten disminuir la redundància en el codi html. Ofereixen la funcionalitat de dissenyar parts comunes en totes les pàgines jsp (com per exemple menús, imatges, el logotipus...) en pàgines separades i, posteriorment, afegir-les inserint etiquetes d'aquesta estructurals.

Per tal de poder utilitzar aquesta capacitat, el desenvolupador haurà de crear un fitxer de configuració (tiles-defs.xml) al qual s'associï a cada pàgina que conté una part de codi comú un nom amb el qual s'identificarà des de l'etiqueta estructural.

Un exemple podria ser l'etiqueta <tiles:insert attribute="pàgina.jsp">, la qual substituiria a la tradicional etiqueta jsp <%@ include file="pàgina.jsp"%>.

Per més informació sobre les llibreries vegeu [5] o les tld del framework Struts.

B.4.3.2 ActionForm o FormBean

La classe "ActionForm[34]" es pot definir com la representació d'un formulari dins del framework Struts. Proporciona un conjunt d'atributs, corresponents amb les dades del formulari, conjuntament amb els mètodes necessaris per accedir als seus valors.

Les seves característiques principals són les següents:

1. El constructor serà nul.
2. Per cada camp del formulari es crearà una propietat (atribut)[35]. A més a més, per cada propietat "xx" existent, "l'ActionForm" haurà de proporcionar un mètode getXx(), que retornarà un "String" amb el valor corresponent, i un mètode setXx(String), que inicialitzarà l'atribut.
3. En el moment que el desenvolupador subclassifiqui aquesta classe no és necessari que sobreescrigui cap mètode, ja que un "ActionForm" no conté lògica de tractament de dades, únicament s'encarrega d'emmagatzemar-les.
4. Si es desitja que el formulari sigui validat, se'n pot sobreescriure el mètode validate() posant-hi tot el codi necessari per comprovar la correctesa de les dades.

[34] Part de la documentació consultada [2] considera aquesta classe part de la capa controlador. No obstant, després d'haver après, entès i utilitzat el framework, s'ha cregut convenient fer la seva explicació dins de la capa vista ja que, "l'ActionForm" es pot definir com la representació d'aquesta dins de la capa controlador.

[35] El valor de l'atribut "property" del camp del formulari i el nom de l'atribut dins de "l'ActionForm" hauran de ser el mateix.

B.5 Flux de la informació

Un cop descrits tots els elements que intervenen en l'arquitectura MVC suportada per Struts podem definir com evoluciona la informació a través de tots ells.

1. L'usuari fa la petició d'un formulari a través d'una URL.
2. El servidor proporciona al navegador la pàgina jsp, construïda amb les etiquetes proporcionades per les llibreries del framework, que conté el formulari.
3. L'usuari omple el formulari i l'envia al servidor, realitzant una petició.
4. La petició és captada per "ActionServlet" (controlador del sistema), el qual determina el "RequestProcessor" a qui delegar les tasques de tractament d'aquesta petició.
5. El "RequestProcessor", un cop rep la petició des "d'ActionServlet", s'encarrega de crear "l'ActionForm" associat a aquesta i omplir-lo amb les dades introduïdes al formulari. Un cop finalitzada aquesta tasca, cerca "l'Action" que satisfà la funcionalitat demanada en la petició, la instancia i n'executa el mètode execute(). Tot això, ho realitza amb l'ajuda del fitxer de configuració "struts-config.xml", del "Digester" i de les classes "ActionMapping" i "ActionForward"
6. Un cop "l'Action" a acabat la seva execució retorna un "ActionForward" amb la informació adient per readreçar el control de l'aplicació (gràcies al fitxer "struts-config.xml").
7. Es retorna la resposta a l'usuari en forma de pàgina jsp.

B.6 Passos a seguir per crear una nova funcionalitat

Un cop explicat el framework Struts i el flux de dades a través del sistema s'intentarà fer una breu descripció dels passos que hauria de seguir un desenvolupador per tal de crear una nova funcionalitat en una aplicació web que treballi amb Struts.

Aquests passos serien els següents:

1. Descàrrega del framework Struts i ubicació d'aquest dins de l'aplicació (en la figura B.6.1 es pot veure com queda l'estructura de directoris en un contenidor Tomcat).
2. Disseny d'un formulari amb les etiquetes necessàries que ofereixen les llibreries del framework
3. Crear un "FormBean" subclassificant "l'ActionForm" que ofereix Struts. Aquesta classes "FormBean" serà la que s'encarregarà de recopilar les dades introduïdes en el formulari.
4. Si és necessari crear "ResultBeans[36]" per tal d'emmagatzemar les dades de resposta del servidor i, posteriorment, presentar-les a l'usuari.
5. Crear una Action la qual s'encarregarà de dur a terme la funcionalitat necessària.
6. Modificar el fitxer de configuració "struts-config.xml" per tal d'unir les següents dades:
    a. El nom dels "FormBean" que s'utilitzaran amb la seva ruta, és a dir, el nom amb el directori (paquet) on es podran trobar aquest "FormBean".
    b. La URL on s'enviaran les dades del formulari amb la ruta de l'Action que les haurà de processar.
    c. "L'Action" amb el nom del "FormBeans" corresponent a la petició.
    d. El resultat que retornarà "l'Action", en forma "d'ActionForward" amb la ruta de l'element al qual se li transferirà el control (ja sigui una pàgina jsp o una altra "Action).
7. Modificar el fitxer de configuració del contenidor (en aquest cas web.xml) per tal de mapejar el controlador ("ActionServlet) i els seus paràmetres inicials ("struts-config.xml).

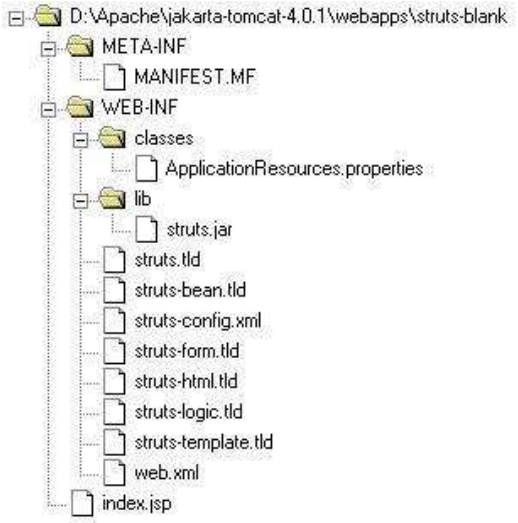


figura B.6.1 (extret de [6]) Estructura de directoris

[36] Aquest tipus de classes no s'han descrit extensament anteriorment ja que podríem dir que equivalen a un "FormBean". No obstant, mentre que un "FormBean" dóna les dades introduïdes per l'usuari al servidor, el "ResultBean" segueix el camí contrari, servint per presentant les dades de resposta del servidor a l'usuari.

B.7 Capacitats addicionals

El framework Struts proporciona tres capacitats bàsiques addicionals que permeten fer una aplicació més consistent, exportable i fiable.

Aquestes funcionalitats, ordenades segons la seva proximitat[37] a l'usuari, són:

1. Internacionalització de la vista
2. Validació de les dades
3. Tractament de les excepcions

A continuació es realitzarà una descripció exhaustiva d'aquestes tres tasques.

B.7.1 Internacionalització de la vista

La capacitat d'internacionalització permet realitzar una aplicació exportable a diferents idiomes sense que això impliqui crear noves vistes (pàgines jsp) amb els missatges en la llengua corresponent.

Struts dóna suport, únicament, a la internacionalització de la vista mitjançant la creació d'un fitxer de propietats, l'utilització de les llibreries d'etiquetes que ofereix i l'ajut de diverses classes.

Les 5 classes relacionades amb la internacionalització d'una aplicació són les següents [2]:

- "Locale": classe que representa un país i una llengua en particular (ja que un mateix país pot tenir diferents llengües, segons el territori on ens situem). A més a més, també indica el format que han de tenir algunes dades, com per exemple, les dates i els números.
- "ResourceBundle": classe situada al paquet "java.util" que proporciona les eines per suportar els missatges en diverses llengües.
- "PropertyResourceBundle": subclasse de "ResourceBundles" que permet tractar els missatges a internacionalitzar com una parella "clau-valor" (tal i com es defineixen en els fitxers de propietats).
- "MessageFormat": classe que permet substituir trossos d'un missatge amb arguments especificats en temps d'execució.
- "MessageResources": classe que es troba a "org.apache.struts.util" i que permet realitzar el tractament de diversos recursos (bases de dades, fitxers de propietats...) per tal de demanar-ne el valor d'un determinat missatge per a un "Locale" concret.

[37] S'entén com a proximitat la transparència en que les tasques descrites es realitzen. És a dir, la internacionalització es realitza directament en el navegador, la validació es pot realitzar en el navegador o en el servidor com veurem més endavant i, finalment, el tractament de les excepcions es realitza en el servidor.

Els passos a seguir per tal d'internacionalitzar l'aplicació són els següents:

1. Crear el fitxer de propietats (en el cas que ens ocupa, MessageResources.properties) i situar-lo al directori corresponent (en aquest cas, el directori serà CATALINA_HOME/WEB-INF/classes)
2. Omplir el fitxer de propietats amb tots els missatges que es creguin necessaris associats a un identificador que ens permetrà accedir-hi des de la pàgina .jsp gràcies a la utilització de les etiquetes proporcionades per les llibreries del framework.
   D'aquesta manera, el contingut del fitxer tindrà l'aspecte següent:
   nom.clau1=missatge1
   nom.clau2=missatge2
   Si volem que els missatges siguin parametritzats només fa falta que els creem de la següent manera:
   nom.clau=missatge {0}
   on {0}serà substituït pel paràmetre corresponent.
3. Mapejar el fitxer de propietats a "struts-config.xml" de la manera següent:
   <message-resources parameter="MessageResources" null="false"/>
4. Carregar la llibreria tags-bean a la pàgina jsp per tal de poder utilitzar l'etiqueta <bean:message>, la qual, com ja s'ha explicat en l'apartat 1.4.3.1.2, conté dos atributs importants:
   .- key: identificador del missatge dins del fitxer propietats (ex: nom.clau1)
   .- arg0 (arg1, arg2...): identificador del paràmetre que s'inclourà al missatge principal.
5. Per tal de fer la internacionalització, només fa falta crear tants fitxers de propietats com llenguatges vulguem tractar, tenint en compte que cada llenguatge té el seu estàndard per al nom del fitxer. Per exemple:
   fitxer_es.properties per a la llengua espanyola
   fitxer_jp.properties per a la llengua japonesa
   Un cop creats els fitxers, el navegador utilitzarà el de la llengua que tingui configurada o, si no existeix, el fitxer per defecte (fitxer.properties).
   També podem indicar l'idioma a utilitzar amb el mètode setLocale() d'Action.

B.7.2 Validació de les dades

La capacitat de validació de les dades permet crear una aplicació més fiable i consistent. A més a més, també permet detectar possibles errors molt abans del que es faria sense una validació i, d'aquesta manera, estalviar temps i capacitat de processament.

La validació en el framework Struts es por realitzar de dues maneres:

1. Validació manual (s'executa en el servidor):
    a. Fent que les "Actions" que reben les dades dels formularis controlin tots els possibles errors que es poden donar abans de treballar sobre el model. Si detecten algun error, s'enviarà a l'usuari la vista corresponent indicant què ha passat
    b. Fent una validació als "FormBeans" utilitzant el mètode validate(). Aquesta validació permet que, quan es troba un error, el formulari es torni a presentar a l'usuari (indicant on hi ha el problema) sense que s'hagi arribat a cridaren cap moment a cap Action
2. Validació automàtica utilitzant totes les eines que ofereix struts. Aquest tipus de validació es pot executar tant en el servidor (si el navegador no té activat javascript) com en el client (en cas que javascript estigui operatiu en el navegador).
   La validació automàtica es basa en dos fitxers .xml:
   .- "validator-rules.xml": fitxer que tipifica tots els possibles errors que es poden donar en un formulari i indica quines classes del framework seran les encarregades de detectar-los i tractar-los.
   .- "validation.xml": fitxer de configuració on el desenvolupador indica per cada formulari les normes de validació que han d'existir.
   D'altra banda, per tal que aquesta validació es pugui dur a terme, el "FormBean" creat haurà de ser subclasse de "ValidatorForm", en comptes "d'ActionForm", i en el seu mètode validate() haurà de fer una crida al mateix mètode de la superclasse.
   Si es detecta algun error, el formulari es presentarà de nou a l'usuari. Per tal d'indicar on sha produït el problema és necessari utilitzat l'etiqueta <html:errors> de la llibreria Html, de tal manera que struts detectarà l'error i presentarà en el lloc d'aquesta etiqueta el missatge que s'indiqui a validation.xml[38].

   Si es desitja que la validació es faci mitjançant javascript, s'haurà d'afegir el següent codi a la pàgina .jsp:

   ```
   <html:javascript formName="FormBean"/> a qualsevol lloc
   <html:form onsubmit="return validateFormBean(this);">
   ```

   Finalment, s'haurà d'indicar a struts-config.xml que la validació serà automàtica de la següent manera:

   ```
   <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
     <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
   </plug-in>
   ```

[38] Al fitxer "validation.xml" s'hi especificarà l'identificador del missatge que es desitja presentar d'entre els que estan definits al fitxer de propietats.

A continuació s'indicaran alguns dels errors tipificats a "validator-rules.xml" que s'han utilitzat en el projecte que s'està documentant i com aquest es configuren a "valodation.xml" per a un determinat paràmetre d'un formulari.

Exemples d'errors:

- "required": aquest error ens indica que un determinat camp del formulari és obligatori, és a dir, no pot ser nul. Una possible configuració podria ser aquesta:

```
<form-validation>
  <formset>
    <form name="formulari1">
      <field property="atribut1" depends="required">
        <arg0 key="formulari.atribut1Parameter"/>
        <msg name="required" key="formulari1.nullData"/>
      </field>
    </formset>
  </form-validation>
```

Aspectes importants que cal destacar:

a. El fitxer "validation.xml" té com arrel l'etiqueta <form-validation>
b. La definició dels "FormBeans" i les dades que se'n volen validar es realitzarà dins la secció identificada per l'etiqueta a<formset>
c. Per cada "FormBean" del qual es vulgui realitzar la validació es crearà una secció identificada per l'etiqueta <form>
d. Per cada propietat del "FormBean" que es vulgui validar es crearà una secció identificada per l'etiqueta <field>, en la qual s'hi indicarà:
    i. Propietat a validar (gràcies a l'atribut "property").
    ii. Característica/ques que s'ha/n de validar (gràcies a l'atribut "depends").
    iii. Missatge que sortirà a la vista en cas d'haver-hi error (a l'etiqueta <msg>)
    iv. Paràmetres necessaris per construir el missatge d'error (a l'etiqueta <argX>).

En l'exemple que s'ha posat anteriorment passaria el següent: si la propietat "atribut1" del FormBean "formulari1" és nul·la es traurà el missatge d'error identificat al fitxer de propietats amb la clau "formulari1.nullData" i parametritzat amb el contingut identificat per la clau "atribut1.Parameter".

- "validwhen": aquest error ens indica que un determinat camp del formulari només serà vàlid depenent del seu propi valor o dels valors d'altres camps del mateix formulari. Una possible configuració podria ser aquesta:

```
<form-validation>
  <formset>
    <form name="formulari1">
      <field property="atribut1" depends="validwhen">
```

```
        <msg name="validwhen" key="formulari1.notValid"/>
        <var>
          <var-name>test</var-name>
          <var-value>(*this*!=”-“)</var-value>
        </var>
       </field>
     </formset>
   </form-validation>
```

Aquest exemple, el que indica és que la propietat “atribut1” del FormBean “formulari1” només serà vàlida en cas que el seu valor sigui diferent d’un guió (-).

- “date”: aquest error ens indica que un determinat camp del formulari ha de ser una data ben formada segons el format establert. Una possible configuració podria ser la següent:

```
<form-validation>
  <formset>
    <form name="formulari1">
      <field property="atribut1" depends="date">
        <msg name="date" key="formulari1.incorrectDate"/>
          <var>
          <var-name>datePatternsStrict</var-name>
          <var-value>dd/MM/yyyy</var-value>
        </var>
      </field>
    </formset>
  </form-validation>
```

Aquest exemple, el que indica és que la propietat “atribut1” del FormBean “formulari1” només serà correcta en el cas que sigui una data i el seu format sigui del tipus 31/05/2006.

- “mask”: aquest error ens indica que un determinat camp del formulari només serà correcte si compleix un format adequat. Una possible configuració podria ser la següent:

```
<form-validation>
  <formset>
    <form name="formulari1">
      <field property="atribut1" depends="mask">
        <msg name="mask" key="formulari1.incorrectFormat"/>
          <var>
          <var-name>mask</var-name>
          <var-value>^\d{4}$</var-value>
        </var>
      </field>
    </formset>
  </form-validation>
```

Aquest exemple, el que indica és que la propietat “atribut1” del FormBean “formulari1” només serà correcta en el cas que sigui un número de 4 dígits.
Per més informació sobre validació vegeu [2] o el fitxer “validator-rules.xml” d’Struts

B.7.3 Tractament de les excepcions

La capacitat de tractament de les excepcions permet crear una aplicació més fiable i consistent i avisar a l'usuari dels problemes ocorreguts.

Struts ofereix la possibilitat de realitzar el seguiment i tractament de les excepcions que poden sorgir en una aplicació web mitjançant la classe "ExceptionHandler" (situada al paquet "org.apache.struts.action").

Poden distingir dos tipus d'excepcions:

1. Generals: tindran el mateix tractament es donin quan es donin
2. Específiques: seran tractades de manera diferent segons "l'Action" que les hagi provocat.

El control bàsic que el framework realitza sobre les excepcions consisteix en mapejar-les al fitxer "struts-config.xml". D'aquesta manera, es pot fer encaminar la vista cap a una pàgina jsp que informi a l'usuari del que ha passat.
Per tal de mapejar les excepcions en el fitxer de configuració s'utilitza l'etiqueta <exception> en la qual s'indicarà el tipus, o classe, de l'excepció (atribut "type"), la pàgina jsp que s'ha de presentar ("path"), el manegador d'excepcions en cas necessari ("handler") i l'identificador del missatge que es vol presentar a l'usuari ("key").

Segons quin tipus d'excepció estiguem tractant, l'etiqueta <excepcion> s'inclourà en el bloc <global-exceptions> (per a excepcions generals) o en el bloc <action> (per a excepcions específiques).

Per tal de realitzar el tractament d'una excepció amb la finalitat de poder deixar el model en un estat de consistència i sense errors derivats de l'execució del codi que l'ha provocat, el framework proporciona la facilitat de crear els manegadors d'excepcions. Aquest manegadors compliran les següents característiques:

a. Hauran de ser subclasses d'ExceptionHandler i tindran el comportament que el desenvolupador cregui necessari.
b. Seran mapejats en l'atribut "handler" de l'etiqueta <exception>, com ja hem dit amb anterioritat.

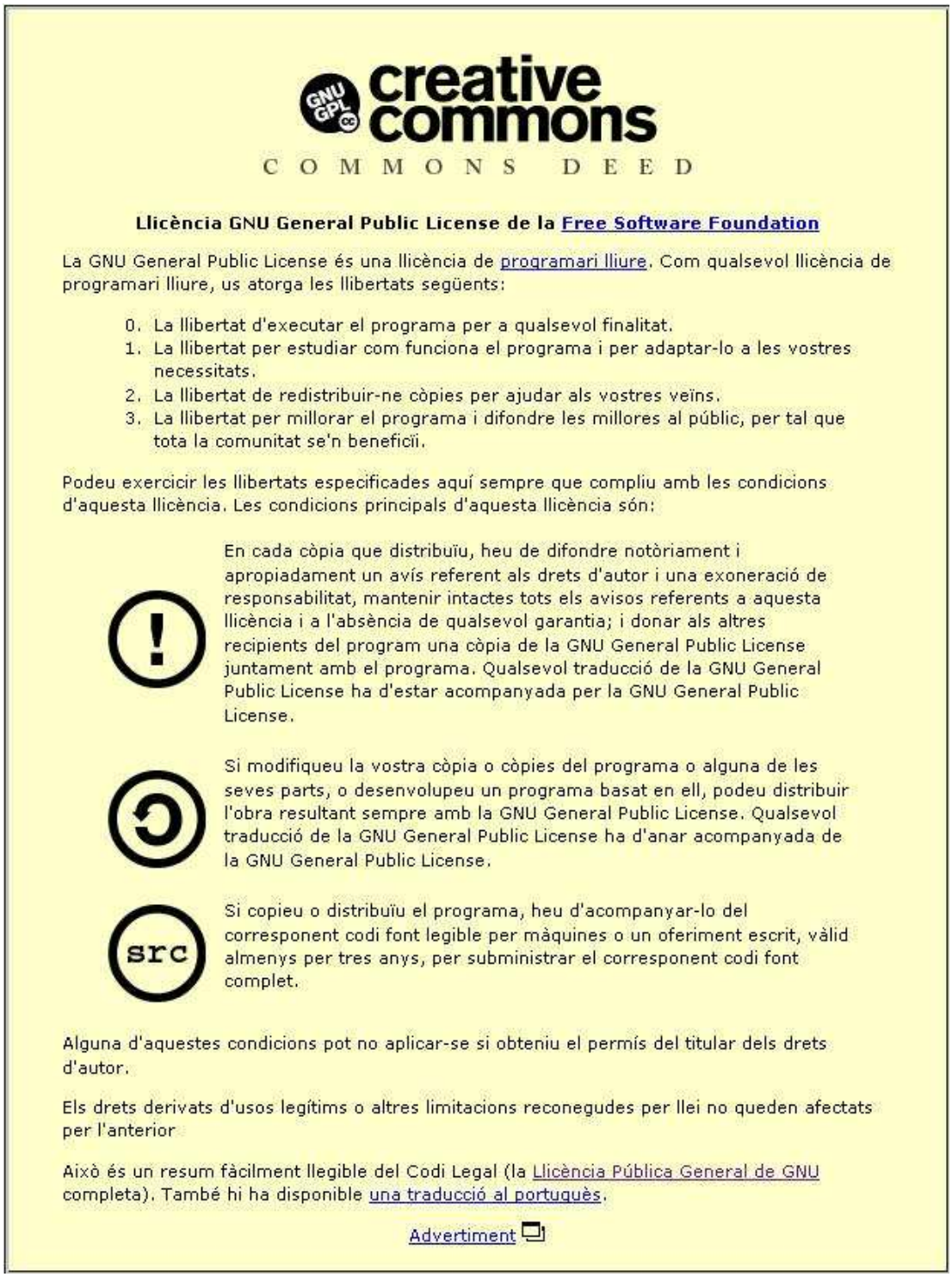
Per més informació sobre el tractament de les excepcions vegeu [2].

Annex C

Llicència

A l'aplicació i documentació realitzades com a treball de final de carrera se'ls hi ha posat la llicència de codi lliure CC-GPL. L'explicació, en versió "Human Readable" i català, es troba en la figura C.1.

D'altra banda, se'n pot llegir la versió completa a la pàgina web: http://www.gnu.org/copyleft/gpl.html.


creative commons

COMMONS DEED

Llicència GNU General Public License de la Free Software Foundation

La GNU General Public License és una llicència de programari lliure. Com qualsevol llicència de programari lliure, us atorga les llibertats següents:

0. La llibertat d'executar el programa per a qualsevol finalitat.
1. La llibertat per estudiar com funciona el programa i per adaptar-lo a les vostres necessitats.
2. La llibertat de redistribuir-ne còpies per ajudar als vostres veïns.
3. La llibertat per millorar el programa i difondre les millores al públic, per tal que tota la comunitat se'n beneficiï.

Podeu exercicir les llibertats especificades aquí sempre que compliu amb les condicions d'aquesta llicència. Les condicions principals d'aquesta llicència són:

En cada còpia que distribuïu, heu de difondre notòriament i apropiadament un avís referent als drets d'autor i una exoneració de responsabilitat, mantenir intactes tots els avisos referents a aquesta llicència i a l'absència de qualsevol garantia; i donar als altres recipients del program una còpia de la GNU General Public License juntament amb el programa. Qualsevol traducció de la GNU General Public License ha d'estar acompanyada per la GNU General Public License.

Si modifiqueu la vostra còpia o còpies del programa o alguna de les seves parts, o desenvolupeu un programa basat en ell, podeu distribuir l'obra resultant sempre amb la GNU General Public License. Qualsevol traducció de la GNU General Public License ha d'anar acompanyada de la GNU General Public License.

Si copieu o distribuïu el programa, heu d'acompanyar-lo del corresponent codi font legible per màquines o un oferiment escrit, vàlid almenys per tres anys, per subministrar el corresponent codi font complet.

Alguna d'aquestes condicions pot no aplicar-se si obteniu el permís del titular dels drets d'autor.

Els drets derivats d'usos legítims o altres limitacions reconegudes per llei no queden afectats per l'anterior

Això és un resum fàcilment llegible del Codi Legal (la Llicència Pública General de GNU completa). També hi ha disponible una traducció al português.

Advertiment

figura C.1 Versió "Human Readable" i en català de la llicència CC-GPL

Annex D

API

A continuació es vol presentar un molt breu mostreig de l'API realitzat sobre el codi de l'aplicació per tal de facilitar-ne l'aprenentatge per a futures ampliacions.

En les figures D.1, D.2 i D.3 es presenta l'API de la classe

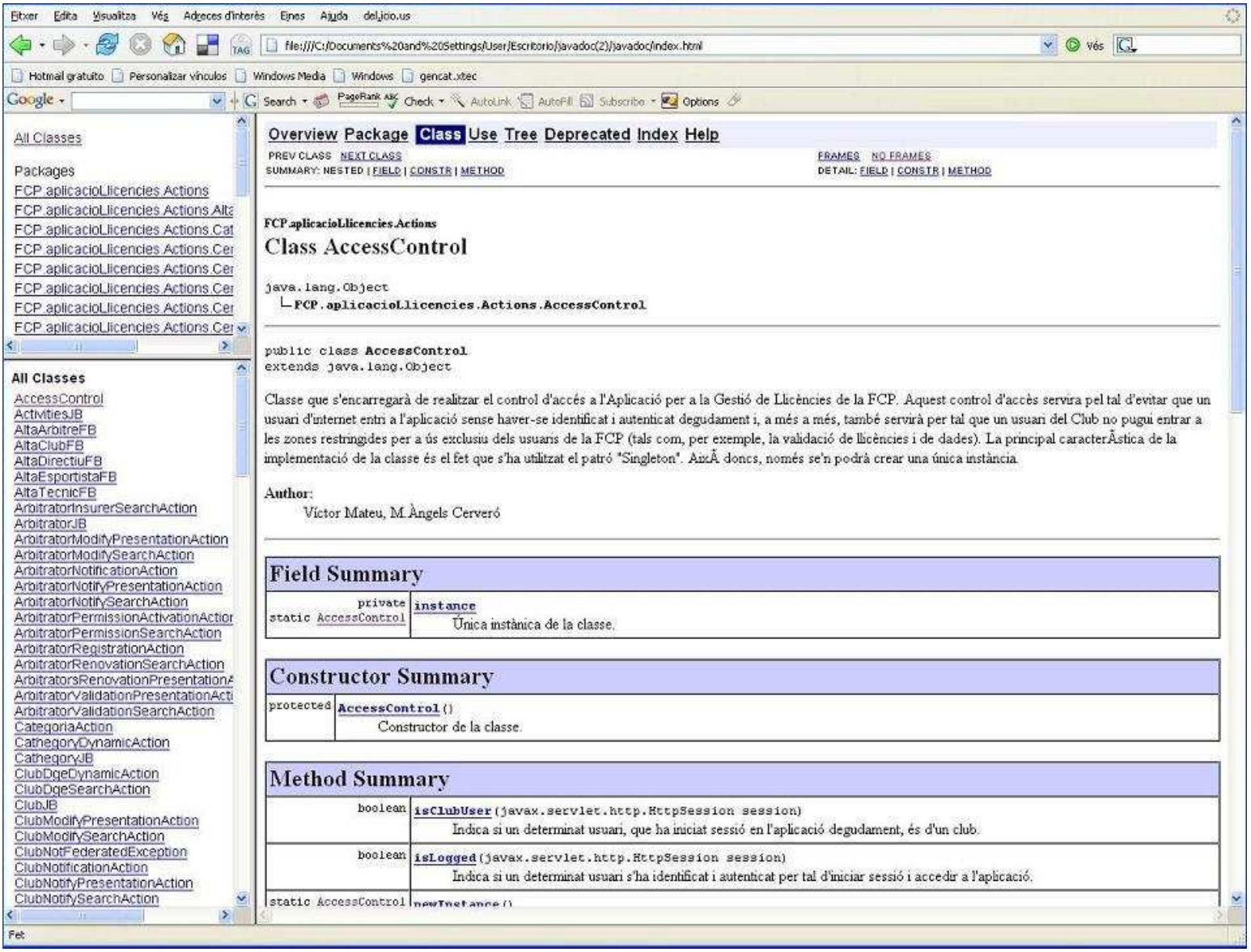


figura D.1 Exemple de l'API

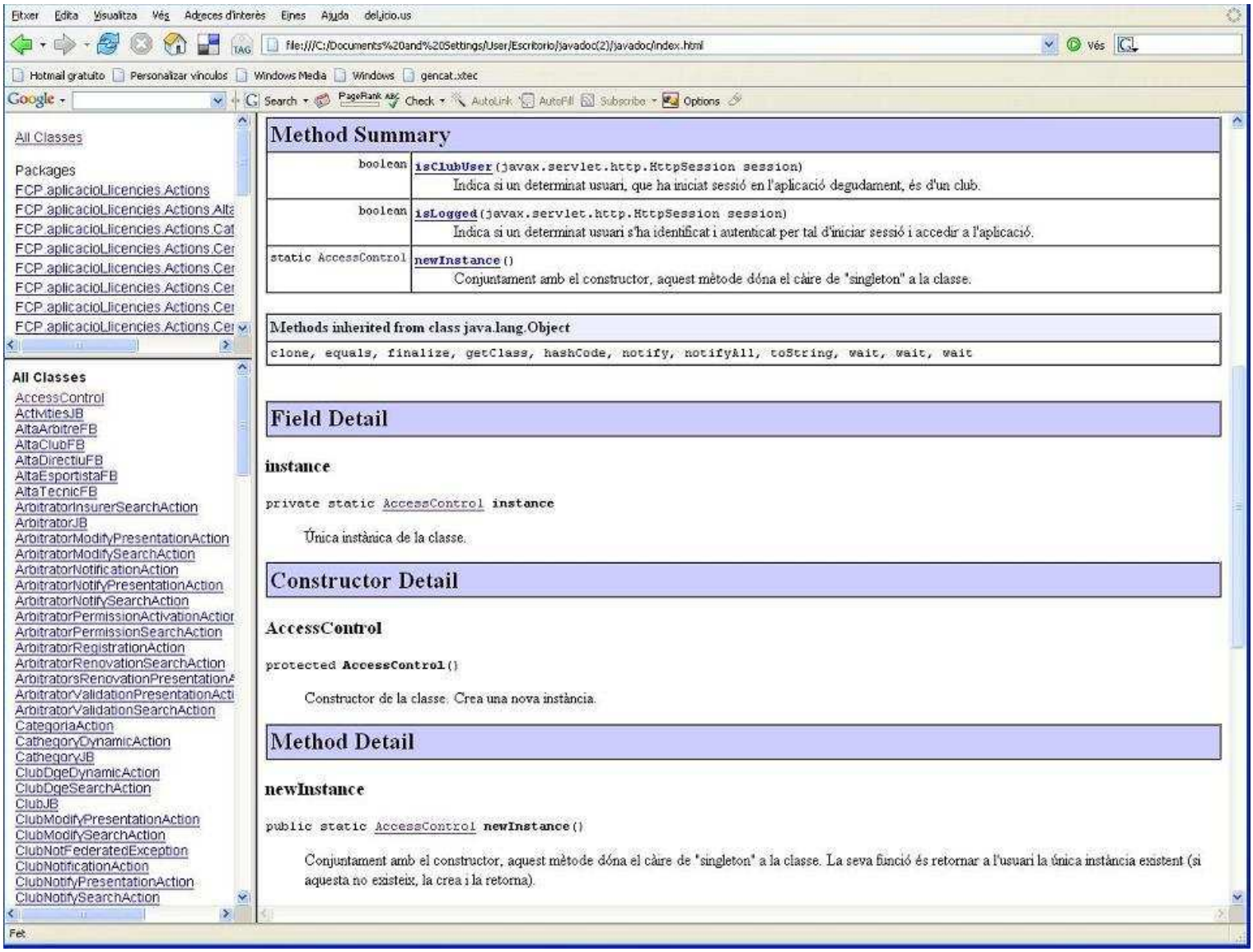


figura D.2 Exemple de l'API

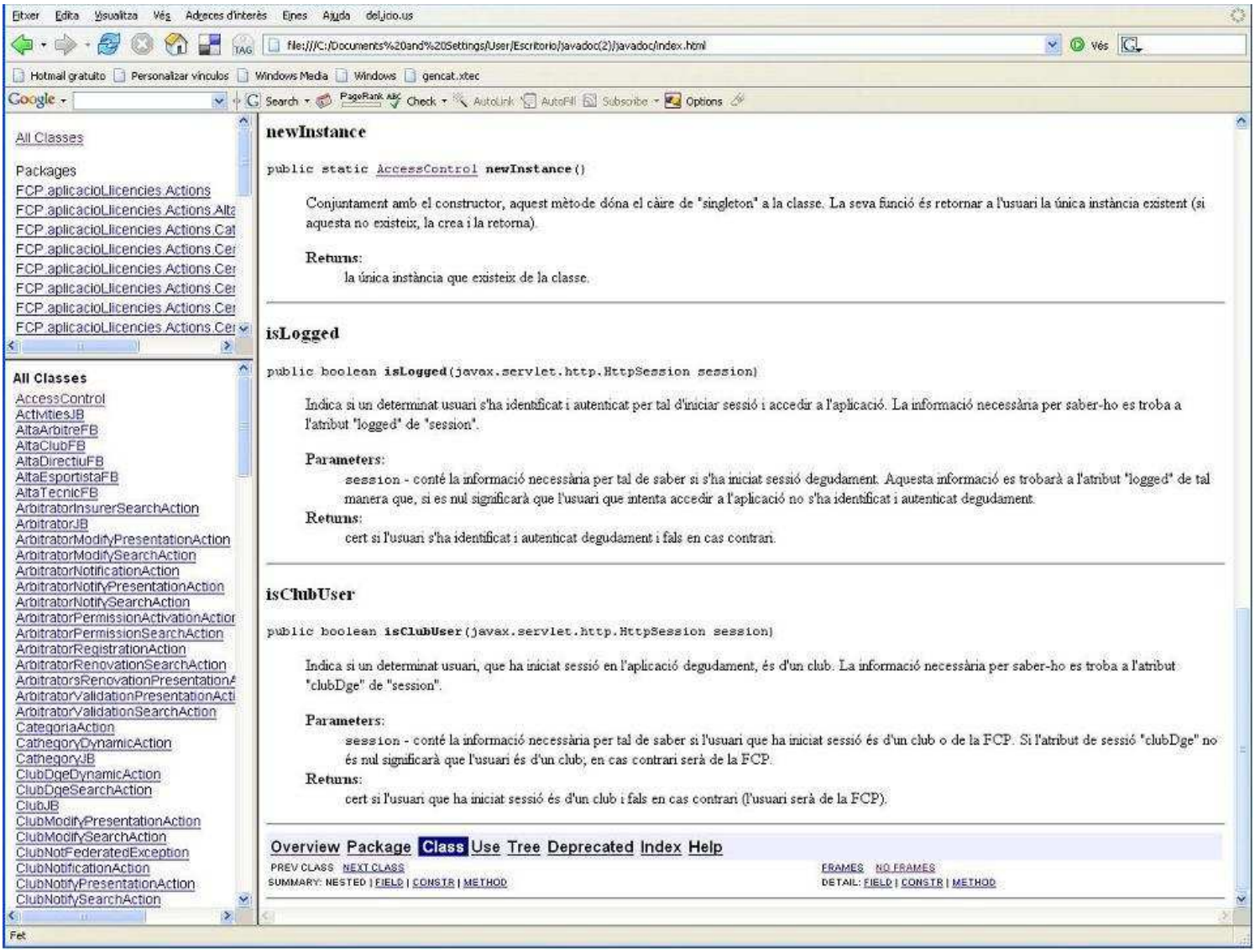


figura D.3 Exemple de l'API

Bibliografia

[1] ONJAVA.COM (9 de novembre del 2001). *Introduction to Jakarta Struts Framework*: Sue Spielman. [Consultat: 16 de setembre del 2006]. Disponible a internet: http://www.onjava.com/pub/a/onjava/2001/09/11/jsp_servlets.html.

[2] THE APACHE SOFTWARE FUNDATION (18 de setembre del 2006). *Struts: User and Developer Guides*. [Consultat: setembre del 2006]. Disponible a internet: http://struts.apache.org/1.2.9/index.html.

[3] THE APACHE SOFTWARE FUNDATION (18 d'octubre del 2004). *DTD for the Struts Application Configuration File (v. 1.2)*. [Consultat: setembre del 2006]. Disponible a: http://struts.apache.org/dtds/struts-config_1_2.dtd.

[4] THE APACHE SOFTWARE FUNDATION (18 de setembre del 2006). *Struts*. [Consultat: setembre del 2006]. Disponible a internet: http://struts.apache.org.

[5] ONJAVA.COM (30 de Juliol del 2003). *Getting the Most Out of the Tag Libraries*: Chuck Cavaness. [Consultat: setembre del 2006]. Disponible a internet: http://www.onjava.com/pub/a/onjava/2003/07/30/jakartastruts.html.

[6] JAVA BOUTIQUE. *Stepping through Jakarta Struts*: Keld H. Hansen. [Consultat: 17 de setembre del 2006]. Disponible a internet: http://javaboutique.internet.com/tutorials/Struts/.

[7] WIKIPEDIA LA ENCICLOPEDIA LIBRE. *Programación por capas*. [Consultat: 19 de setembre del 2006]. Disponible a internet: http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles.

[8] UNIVERSIDAD SIMÓN BOLÍVAR (21 de maig del 2001). *Arquitectura de capas*: Alejandro Teruel. [Consultat: 18 de setembre del 2006]. Disponible a internet: http://www.ldc.usb.ve/~teruel/ci3715/clases/arqCapas.html.

[9] LARMAN, C (2003).*UML i Patrones*. 2a edició ed. Pearson Educación, S.A. ISBN: 84-205-3438-2

[10] GAMMA, E.; HELM, R.; JOHSON, R.; VLISSIDES, J. (2003). *Patrones de Diseño*. 1a edició ed. Pearson Educación, S.A. ISBN: 84-7829-059-1

[11] GIMENO, J.M (2005). *Patrons de disseny GoF: Herència versus Delegació*. Apunts de l'assignatura AES de tercer curs d'ETIG de l'EPS (UdL).

[12] W3C (7 de setembre del 2006). *HyperText Markup Language (HTML)* [Consultat: setembre del 2006]. Disponible a internet: http://www.w3.org/MarkUp/

[13] W3C. *Cascadig Stype Sheets, level 1*: Hakon Wium Lie, Bert Bos. [Consultat: setembre del 2006]. Disponible a internet: http://www.w3.org/TR/REC-CSS1

[14] JAVA SUN (24 d'abril del 2002). *Java Server Pages Technology*: Stephanie Bodoff. [Consultat: setembre del 2006]. Disponible a internet: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro.html

[15] RIBÓ, J.M. (2005). *The Web layer (sections 5.6-5.7)*. Apunts de l'assignatura TAP de tercer curs d'ETIG de l'EPS (UdL).

[16] GIMENO, J.M. (2005). *Una intoducció a XML*. Apunts de l'assignatura TAP de tercer curs d'ETIG de l'EPS (UdL).

[17] PERDRIX, F (2005). *Laboratori d'Iniciació a les Bases de Dades*. Apunts de l'assignatura IBD de segon curs de l'EPS (UdL).

[18] THE APACHE SOFTWARE FUNDATION. *Apache Tomcat*. [Consultat: 19 de setembre del 2006]. Disponible a internet: http://tomcat.apache.org/

[19] RIBÓ, J.M. (2005). *The Web layer (sections 5.4-5.5)*. Apunts de l'assignatura TAP de tercer curs d'ETIG de l'EPS (UdL).

[20] GIMENO, J.M. (2005). *Java i XML*. Apunts de l'assignatura TAP de tercer curs d'ETIG de l'EPS (UdL).

[21] GIMENO, J.M. (2005). *XSL: XML Stylesheet Language*. Apunts de l'assignatura TAP de tercer curs d'ETIG de l'EPS (UdL).

[22] Universidad Simón Bolívar (1 de març del 2001). *COCOMO II: una Familia de Modelos de Estimación*: Alejandro Teruel. [Consultat: setembre del 2006]. Disponible a internet: http://www.ldc.usb.ve/~teruel/ci4713/clases2001/cocomo2.html.

[23] World Wide Web Consortium (21 de setembre del 2006). *Leading the Web to Its Full Potential...*. [Consultat: setembre del 2006]. Disponible a internet: http://www.w3.org/.